A HYBRID ALGORITHM FOR DATA COMPRESSION USING HUFFMAN & GENETIC ALGORITHMS

A Thesis Submitted to the Council of the Science College of Babylon University in Partial Fulfilment of the Requirements

of the Degree of M.SC.

In Computer Science

By

Azher Razak Hadi Witwit

Safar -١٤٢٧ **March - ٢٠٠٦**

خوارزمية هجينة لضغط البيانات بأستخدام Huffman

و الخوارزمية الجينية

رســـالة

مقدمة الى مجلس كلية العلوم - جامعة بابل

وهي جزء من متطلبات نيل درجة ماجستير في علوم الحاسبات

من

أزهر رزاق هادي وتوت

آذار- ٢٠٠٦ صفر-١٤٢٧

Acknowledgments

I am indebted to **Allah** for helping me always to finish what I start and for helping me to present this work.

I would like to express my deep gratitude and appreciation to my supervisor **Dr. Tawfiq A. Al-Asadi** for the advice, support and encouragement he gave me during the course of this work. His suggestions, guidance and moral support in difficult times have been essential for completing this work.

I would like to express my deep gratitude and appreciation to my supervisor **Dr. Abbas Al-Himyari** for support and encouragement for me.

My deep appreciation goes to the Department of Computer Science, head of the department and members of the staff.

I also wish to thank my **special friends** for their cooperation ,and **all the persons** that helped me to finsh this research.

Supervisor Certification

I certify that this thesis was prepared under my supervision at the Department of Computer Science/College of Science/Babylon University, by **Azher R. H. Witwit** as partial fulfilment of the requirements for the degree of Master of Science in computer science.

Signature :
Name : **Dr. Tawfiq A. Abbas Al-Asadi**
Title : **Assistant Professor**
Date : / / ٢٠٠٦

In view of the available recommendation , I forward this theses for debate by the examination committee.

Signature :
Name : **Dr. Abbas M. AL-Bakry**
Title : **Head of Department of Computer Science.**
Date : / / ٢٠٠٦.

Certification of the Examination Committee

We chairman and members of the examination committee, certify that we have studied the thesis entitled **(A HYBRID ALGORITHM FOR DATA COMPRESSION USING HUFFMAN & GENETIC ALGORITHMS)** presented by the student **Azher R.H. Witwit** and examined him in its contents and in what is related to it, and we have found it worthy to be accepted for the degree of Master of Science in computer science with **Very Good** degree.

Signature :
Name **: Dr. Nabeel H. Kaghed**
Title : **Professor**
Date : / / ٢٠٠٦.

(chairman)

Signature :
Name:**Dr. Sattar B. Sadkan**
Title : **Professor**
Date : / / ٢٠٠٦.

(member)

Signature :
Name **: Esraa H. Ali**
Title : **Assistant Professor**
Date : / / ٢٠٠٦.

(member)

Signature :
Name:**Dr. Tawfiq A. Abbas Al-Asadi**
Title : **Assistant Professor**
Date : / / ٢٠٠٦.

(supervisor)

Signature :
Name : Uoda

Title :

Date : / /

٢٠٠٦.

(Dean of college of science-Babylon university)

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

﴿نَرْفَعُ دَرَجَاتٍ مَنْ نَشَاءُ وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ﴾

صدق الله العظيم

سورة يوسف–آية (٧٦)

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

To the soul of my Father….

Το μψ Μοτηερ □.Το μψ Ωιφε □.

Το μψ χηιλδρεν Μυσταφα & Φατεμα □

ωιτη λοωε.

Abstract

Text compression plays an important role and it is an essential object to decrease storage size and increase the speed of data transmission through communication channels .

In this research , a hybrid compression system is introduced which depends on the genetic algorithm for finding the best Huffman tree and finally finding the best compression ratio and then applying an additional compression method on the results of the primary compression by applying the compression method named Oring bits , and also using a decompression algorithms for both Huffman and Oring bits.

The variety in the text characters leads to the variety in the Huffman trees and finally obtaining the best possible compression . In addition, the increase in the frequencies of the text has an affect on the compression rate .The variance has been taken as a fitness function.

The proposed method used one of the crossover operators that depends on the permutation that is cycle crossover (CX) because of the nature of the

problem under research and what it results of variety . The tournament selection has also been used for what it introduces of solution that converge at first slowly to the optimal solution and then the pressure (speed) increases at the end to reach to the optimal solution.

The genetic algorithms has been used where these algorithms research in the search space for the optimal solution and with a method not depending on the application area and without the need for using knowledge of the area.

In the proposed system, the initial population is generated from the individuals, building a Huffman tree for each individual in the population, computing a fitness function for each individual and then selecting randomly two individuals from the population for stepping into the lake of crossover. After that, the new individuals (offspring) are evaluated and compared with the population in order to take a decision of adding them or not.

The selection, crossover and evaluate operation are continued until it reaches the best Huffman tree that represents the best compression for the text.

الخلاصة

يلعب ضغط النصوص دوراً مهماً ويعتبر هدفاً أساسياً لتقليل حجم الخزن وزيادة سرعة تناقل البيانات عبر قنوات الأتصالات .

في هذا البحث نقدم نظام ضغط هجين يعتمد على الخوارزمية الجينية لأيجاد أفضل شجرة Huffman وبالتالي أفضل نسبة ضغط بطريقة Huffman وتطبيق ضغط إضافي على نتائج الضغط الأولي عن طريق تطبيق خوارزمية Oring bits , وأستخدام خوارزمية فك ضغط لكل من Oring bits و Huffman .

أن التنوع في رموز النص يؤدي الى تنوع أشجار Huffman وبالتالي الحصول على أفضل نسبة ضغط ممكن ، بالأضافة الى ذلك فأن زيادة التكرارات في النص لها تأثير على نسبة الضغط .

لقد جرى أعتماد التباين Variance كمقياس لدالة الصلاحية Fitness function ,

وجرى أستخدام أحد عوامل التزاوج التي تعتمد على التباديل Permutation وهو (CX) cycle crossover وذلك بسبب نوع المشكلة ولما ينتجه من تنوع ، وكذلك جرى أستخدام أنتقاء المجاميع Tournament Selection وذلك لما يقدمه من حلول تقترب في البداية ببطى الى الحل الأمثل ويزداد الضغط (السرعة) عند النهاية للوصول الى الحل الأمثل .

لقد جرى أستخدام الخوارزميات الجينية ، حيث تبحث هذه الخوارزميات في فراغ الحلول عن الحل الأمثل وبطريقة غير معتمدة على مجال التطبيق وبدون الحاجة الى أستخدام معرفة خاصة بالمجال .

في النظام المقترح يجري توليد مجتمع أبتدائي من الأفراد ويتم بناء شجرة Huffman لكل فرد في المجتمع ، ويتم أحتساب دالة الصلاحية لكل أفراد المجتمع ، ثم بعد ذلك يتم أختيار فردين عشوائياً من المجتمع للدخول في بحيرة التزاوج ومن ثم يجري تقييم الأفراد الجدد (الأبناء) ومقارنتهم مع المجتمع لأتخاذ القرار بأضافتهما اليه أم لا ، تستمر عملية الأنتقاء والتزاوج والتقييم لحين الحصول على افضل شجرة Huffman والتي تمثل أفضل ضغط للنص .

١.١ Introduction

Data transmission and storage cost money. The more information being dealt with, the more it costs. In spite of this, most digital data are not stored in the most compact form. Rather, they are stored in whatever way make them easiest to use, such as: ASCII text from word processors, binary code that can be executed on a computer, individual samples from a data acquisition system, etc. Typically, these easy-to-use encoding methods require data files about twice as large as actually needed to represent the information. Data compression is the general term for the various algorithms and programs developed to

address this problem. A compression program is used to convert data from an easy-to-use format to one optimized for compactness. Likewise, a decompression program returns the information to its original form [٢٥,٢٩] .

So **Data compression** can be defined as a technique which reduces the size of a file or data collection without affecting the information contained in the file. A compressed file, obviously, requires less storage space. Moreover, electronic transfer of a compressed file is faster than the transfer of an uncompressed file and a smaller file is consequently less susceptible to transmission errors [٢٥].

:;Aي

Finding the optimal way to compress data with respect to resource constraints remains one of the most challenging problems in the field of data compression.

There are two data compression strategies: lossy and lossless. In this research we used the second strategy.

A lossless data compression algorithm takes a string of symbols (typically ASCII characters or bytes) and translates it *reversibly* into another string, one that is *on the average* of shorter length. The words “on the average” are crucial; it is obvious that no reversible algorithm can make all strings shorter, there just aren’t enough short strings to be in one-to-one correspondence with longer strings. Compression algorithms are possible only when, on the input side, some strings, or some input symbols, are more common than others. These can then be encoded in fewer bits than rarer input strings or symbols, giving a net average gain.

There exist many, quite different, compression techniques, corresponding to different ways of detecting and using departures from equiprobability in input

strings. In this research the Huffman method has been used. The idea behind Huffman coding is simply to use shorter bit patterns for more common characters. We obtain different Huffman codes for the characters but sometimes we do not want the code word lengths to vary widely from one symbol to another (example fixed rate channels).

We can make this idea quantitative by considering the concept

of *entropy*. Suppose the input alphabet has *Nch* characters, and that these occur in the input string with respective probabilities *pi, i* = ١, . . .,*Nch,* so that $\sum pi = ١$. And suppose that ***avg*** represents the average size which is equal to $\sum \boldsymbol{pi} * \boldsymbol{codeword\ \ length}_i$, then the variance is equal to

$\sum pi * ((\ \boldsymbol{codeword\ length}_i\ \ )- \boldsymbol{avg})^{٢}$, where the variance of a code measures how much the sizes of the individual codes deviate from the average size.

١.٢ *Basic Concepts in Data Compression*

In this section the basic concepts of data compression are shown:-

١.٢.١ Entropy Coding

We can define the entropy of a signal symbol a_i as $-P_i \log_٢ P_i$. This is the smallest number of bits needed, on the average, to represent the symbol.

The amount of information contained in one, base-n symbol is:

This quantity is called the entropy of the data being transited.

$$H = -\sum_{i=٠}^{n-١} p_i \log_٢ p_i \qquad .....(١.١)$$

The entropy of the data depends on the individual probabilities P_i, and is smaller when all n probabilities are equal [٢٦,٢٨].

١.٢.٢ Prefix Property

The prefix code can be defined as no codeword is a prefix to any other codeword. All prefix codes are uniquely decodable. They can be decoded

instantaneously since the end of the codeword is immediately recognizable [٨,١٣].

Example:

A: ١٠

B: ٠٠

C: ٠١١

D: ١١١

ABADCB -> ١٠٠٠١٠١١١٠١١٠٠ -> ABADCB

Prefix codes are based on the following facts:

١. In an optimal code, shorter code words should be used for more frequent symbols.

٢. In an optimal code, the two less frequent symbols have code words with the same length.

١.٢.٣ Variance

The variance of a code is defined as a measurement of how much the size of the individual codes deviates from the average size [٨].

$$\text{Variance} = \sum_{i=1}^{n} p_i (a_i - A)^2 \quad \ldots\ldots(١.٢)$$

where,

p: probability

a_i: number of bits(length code word)

A: average size

١.٣ Data Compression Strategies

Table (١-١) shows two different ways that data compression algorithms can be categorized. The methods have been classified as either **lossless** or **lossy**. A **lossless** technique means that the restored data file is identical to the original. This is absolutely necessary for many types of data, for example: executable code, word processing files, tabulated numbers, etc. You cannot afford to misplace even a single bit of this type of information. In comparison, data files that represent images and other acquired signals do not have to be kept in perfect condition for storage or transmission. All real world measurements inherently contain a certain amount of noise. If the changes made to these signals resemble a small amount of additional noise, no harm is done. Compression techniques that allow this type of degradation are called **lossy**. This distinction is important because lossy techniques are much more effective at compression than lossless methods. The higher the compression ratio, the more noise are added to the data [٢٩,٢٤].

Table(١-١) Data Compression Techniques

Lossless	Lossy
Run-Length Encoding	CS&Q
Huffman	JPEG
Delta	MPEG
LZW	
GOLOMB	

١.٤ Literature Review

There are many researches and literature in the area of compression, but we confide the ones related to our research .Among those literatures are:

WEE WEONG NG and others[٣٤] presented a research on a genetic algorithm for performing lossless compression on text data. The text can be unambiguously parsed into words then can be replaced by code. The genetic algorithm here approximates a solution to the optimal dictionary construction problem. Where each chromosome represents a dictionary, each gene is of length w_i , where $١ \leq w_i \leq$ n . The population consists of a set of chromosomes whose genes are randomly initialised with characters of one length. The fitness function is the coding rate(b/s), where b is the number of bits in the text encoded and s is the number of symbols in the uncoded text. Strong chromosomes are those that yield lower coding rates. The crossover includes selecting,exchanging ,fusion, and diffusion. The selection strategy for choosing parent chromosome is random. The exchange operator exchanges genes between two parents chromosomes, the strategy followed here is that the weak genes of one parent chromose are placed by strong genes of another parent chromosome. The fusion operater fuses two genes into a single gene. The diffusion operator splits genes into smaller genes. Both the fusion and diffusion strategies selects weak gene for fusion and diffusion respectively. The compression algorithm used here is the huffman algorithm.

Jessica Socha [١٧] presented a research the primary purpose of which is to outline and describe the effectiveness of meta-heuristics (genetic algorithms(GA), simulated annealing(SA), tabu search(TA)) for the lossless compression problems. In our research we focused on the genetic algorithm. The compression used is the huffman algorithm. The GA was used to derive an

optimal tree for lossless compression of an alphabet of the English language. This work included a chroosome represntation where each node in the tree would always contain only a single letter. The selection schema based on the roulette wheel and the elitist strategy. The fitness of a chromose is calculated by using redundency of the coding schema it represents. To perform the crossover of two chromosomes, a random subtree from each chromose is selected and swapped. Once a subtree has been selected for swapping, its parents is recorded. The subtree is removed from the tree, and attached to the parent of the opposite subtree. The researcher had used a repair algorithm of the subtree to be swapped before they are attached to the opposite parent to avoid a resulting tree with too few or too many leafs nodes, and possibly dublicated characters. To mutate a chromosome two terminal nodes of a single tree are selected at random and swapped. The GAs optimal parameters were determined after running many test sets. Impirically, the GA shows the best performance with a small population size. The GA also produced better results when it was given more time to run. As a conclusion of the research as a comparative study of meta-heauristics, it was shown that the GA/Tabu hybird is the most effective meta-heauristics for the lossless compression problem.

H. J. Martinez and others[١٤], presented a technique for image compression using binary Evolutionary Cellular Automata (EVA) which was evolved with a Genetic Algorithm (GA). The method is applied to black and white fingerprint images. The Genetic algorithm is guided by a fitness function consisting of a similarity measure between the configurations of the EVA and the target image. When a suitable approximation of the target image is achieved it can be codified with the rule numbers and logical operations linking the EVA. In this way the original image undergoes a compression process since it will be represented by a considerably smaller bit amount. Typical compression rates of the order of ١٠٠:١ are achieved in the experiments. A codified or compressed

image can be regenerated recovering a very good approximation with a typical similarity of ٩٦٪ to the original image. The
method is compared in speed and compression rate with other commercially and widely used compression algorithms.

Andrew Armstrong and Jamming Jiang [٣], described a novel approach to image quantization by using a genetic algorithm to find near-optimal code words. The proposed system applies genetic operators directly on pixel blocks thus avoiding any conversion to a conventional format. To operate on the data this way, operators for fitness, combination and mutation have been tailored to produce near optimal results in terms of high visual and statistical quality. The popular Competitive Learning Neural Network was chosen to benchmark the system, because in terms of format, data and operating it is the closest stochastic technique to the proposed algorithm. The ability for the Genetic Algorithm to reach a near global optimum is demonstrated by the quality of the reconstructed images produced, when compared to the Neural Network for the same compression ratios. A key feature of the Genetic Algorithm is the ability to produce codebooks that contain sufficient words to fully convey detail, flat regions and shading. On the other hand the Competitive Learning Neural Network, when used under the same settings, produces high levels of detail at the cost of heavy artefacts in smooth and shaded areas. When both compressors are measured by PSNR values of their reconstructed image quality, the proposed algorithm constantly outperforms the benchmark for a group of test samples, which is evidenced by our extensive experimental reports.

Al-Asadi Tawfiq A.[٣٠] proposed a compression system based on genetic algorithm (steady state) for finding optimal compression ratio. He described the design and implementation of novel compression and decompression algorithms .

The GA was applied to obtain an optimal order which gives a highest compression ratio,where the compression algorithm treats the data as blocks and the compression ratio was increased through rearranging the order of the blocks.

The proposed compression method and the compression system were examined by using different images with different sizes and they gave good compression ratios, and this depended on image nature and many parameters such as block size, chromosome length and population size .

١.٥ Aim of Research

Finding the best Huffman tree (best compression) ; and we will get on Codewords with approximate lengths (Minimum Variance Huffman codes) ; and this will help in the transmission operation .

١.٦ Thesis Organization

This thesis has been organized into five chapters:-

Chapter One :- Overview.

This chapter gives an introduction to data compression , and also illustrates some concepts in data compression. It gives an overview of data compression strategies and literature review.

Chapter Two:- Compression Techniques.

This chapter describes the compression techniques and precisely huffman method.

Chapter Three:- Genetic Algorithm.

This chapter describes the genetic algorithm and genetic operators.

Chapter Four:- The Proposed compression System.

This chapter describes and discusses the components of the proposed method.

Chapter Five:- Implementation , Conclusions and Future Works.

This chapter presents some experimental results using the proposed method and provides the conclusions of the research.

٢.١ *Introduction*

Finding the optimal way to compress data with respect to resource constraints remains one of the most challenging problems in the field of source coding. Data compression (or source coding) is the process of creating binary representations of data which requires less storage space than the original data [٣٤,١٢].

Figure(٢-١) shows the block diagram of data compression:-

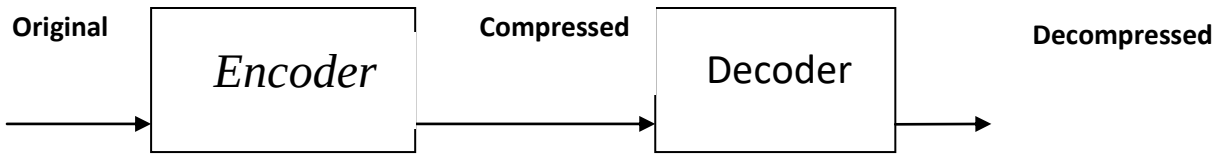


Figure (٢-١) Block Diagram of Data Compression

The reason of compression is:

١- Conserving storage space.

٢- Reducing time for transmission:-

Faster to encode, send, and decode them to send the original.

٣- Progressive transmission:

Some compression techniques allow us to send the most important bits first so we can get a low resolution version of some data before getting the high fidelity version.

٤- Reducing computation:

Use less data to achieve an approximate answer [٩].

٢.٢ *Compression Methods*

There are many compression methods; we will mention some of them.

٢.٢.١ Run-Length Encoding

Data files frequently contain the same character repeated many times in a row. For example, text files use multiple spaces to separate sentences, indent paragraphs, format tables & charts, etc. Digitized signals can also have runs of the same value, indicating that the signal is not changing. For instance, an image of the nighttime sky would contain long runs of the character or characters representing the black background. Likewise, digitized music might have a long run of zeros

between songs. Run-length encoding is a simple method of compressing these types of files [٤,٢٩,١٨,٨].

Figure (٢-٢) illustrates run-length encoding for a data sequence having frequent runs of zeros. Each time a zero is encountered in the input data, two values are written to the output file. The first of these values is a zero; a flag to indicate that run-length compression is beginning. The second value is the number of zeros in the run. If the average run-length is longer than two, compression will take place. On the other hand, many single zeros in the data can make the encoded file larger than the original. Many different run-length schemes have been developed. For example, the input data can be treated as individual bytes, or groups of bytes that represent something more elaborate, such as floating point numbers. Run-length encoding can be used on only one of the characters (as with the zero below), several of the characters, or all of the characters.

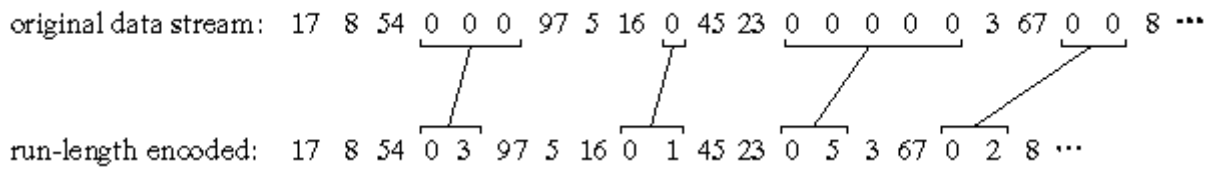


Figure (٢-٢) Run Length Encoding Example

Example of run-length encoding, Each run of zeros is replaced by two characters in the compressed file: a zero to indicate that compression is occurring, followed by the number of zeros in the run.

٢.٢.٢ *Golomb Code*

The Golomb code for non-negative integer n , can be an effective Huffman code. The code depends on the choice of a parameter b . The first step is to compute the two quantities :

$$q=\left\lfloor \frac{n-1}{b} \right\rfloor \quad , \qquad r = n-qb-1 \qquad ....(2.1)$$

Following which, the code is constructed of two parts , the first is the value of q+١, coded in unary , and the second , the binary value of r coded in either bits (for the small remainders) or in $\left\lceil \log_2 b \right\rceil$ bits (for the large ones). $\left\lfloor \log_2 b \right\rfloor$

Choosing b=٣ , e.g. produces three possible remainders ٠,١, and ٢. They are Coded ٠ , ١٠ ,and ١١, respectively . Choosing b=٥ produces the five remainders ٠ through ٤ , which are coded ٠٠,٠١,١٠٠,١٠١,and ١١٠ [٨].

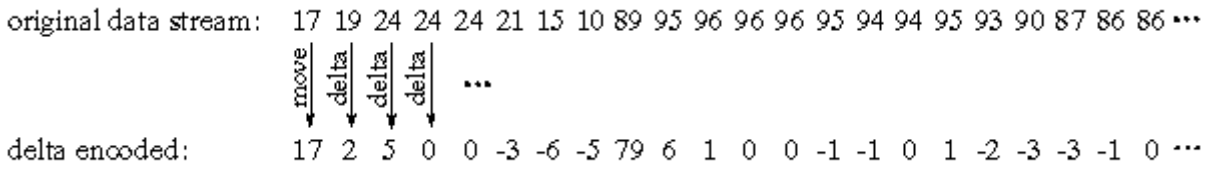
Figure (٢-٣) shows some examples of the Golomb code for b=٣ and b=٥ .

n:	1	2	3	4	5	6	7	8
b=3:	0|0	0|10	0|11	10|0	10|10	10|11	110|0	110|10
b=5:	0|00	0|01	0|100	0|101	0|110	10|00	10|01	10|100

Figure(٢-٣) Golomb Encoding

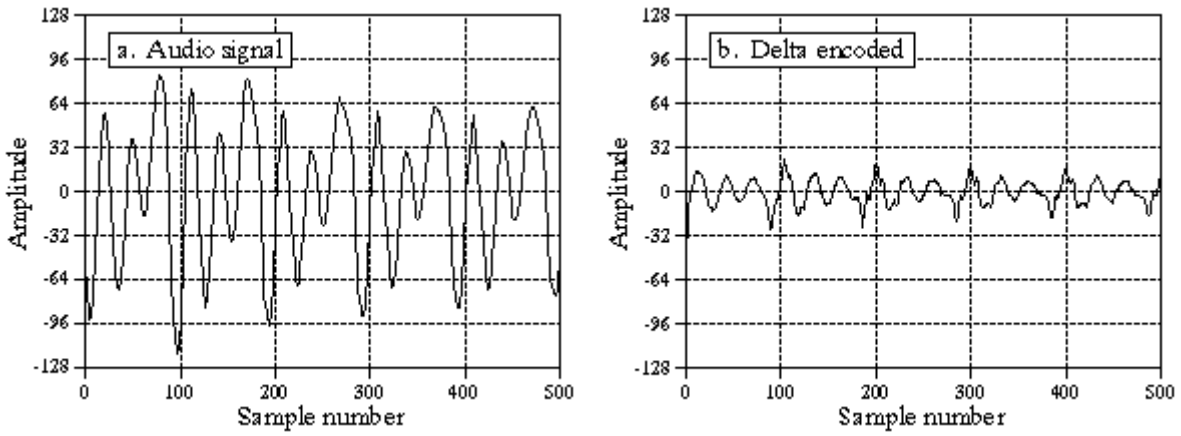
٢.٢.٣ Delta Encoding

In science, engineering, and mathematics, the Greek letter delta is used to denote the change in a variable. The term delta encoding, refers to several techniques that store data as the difference between successive samples (or characters), rather than directly storing the samples themselves. Figure (٢-٤) shows an example of how this is done. The first value in the delta encoded file is the same as the first value in the original data. All the following values in the encoded file are equal to the difference (delta) between the corresponding value in the input file, and the previous value in the input file [٢٩].



Figure(٢-٤) Delta Encoding Example

Delta encoding can be used for data compression when the values in the original data are *smooth*, that is, there is typically only a small change between adjacent values. This is not the case for ASCII text and executable code; however, it is very common when the file represents a signal. For instance, Figure(٢-٥a) shows a segment of an audio signal, digitized to ٨ bits, with each sample between -١٢٧ and ١٢٧. Figure(٢-٥b) shows the delta encoded version of this signal. The key feature is that the delta encoded signal has a lower amplitude than the original signal. In other words, delta encoding has increased the probability that each sample's value will be near zero, and decreased the probability that it will be far from zero. This uneven probability is just the thing that Huffman encoding needs to operate. If the original signal is not changing, or is changing in a straight line, delta encoding will result in runs of samples having the same value.



Figure(٢-٥) Delta Encoded of Signal

Figure (٢-٥a) is an audio signal digitized to ٨ bits. Figure (٢-٥b) shows the delta encoded version of this signal. Delta encoding is useful for data compression if the signal being encoded varies slowly from sample-to-sample.

٢.٢.٤ **LZW** [٢٩,١٨,١٣,١٤]

LZW is named after Abraham Lempel, Jakob Ziv and Terry Welch, the scientists who developed this compression algorithm. It is a lossless 'dictionary based' compression algorithm. Dictionary based algorithms scan a file for sequences of data that occur more than once. These sequences are then stored in a dictionary and within the compressed file, references are put wherever repetitive data occurr.

Savings arise from the fact that pointer and length can be encoded more efficiently than the string it represents. Compression efficiency increases with the length of the text.

٢.٢.٥ *Oring Bits [٨]*

This method starts with a sparse string L_1 of size n_1 bits. In the first step, L_1 is divided into k substrings of equal size. In each substring all bits are logically Ored, and the results (one bit per substring) become string L_2, which will be compressed in step ٢. For example, let substrings of L_1, that we divide into ١٦ substrings of size ٤ each are:

L_1 = ٠٠٠٠ | ٠٠٠٠ | ٠٠٠٠ | ٠١٠٠ | ٠٠٠٠ | ٠٠٠٠ | ٠٠٠٠ | ١٠٠٠ | ٠٠٠٠

| ٠٠٠٠ | ٠٠٠٠ | ٠٠٠٠ | ٠٠١٠ | ٠٠٠٠ | ٠٠٠٠ | ٠٠٠٠

After Oring each ٤-bit substring we get the ١٦-bit string L_2= ٠٠٠١ | ٠٠٠١ | ٠٠٠٠ | ١٠٠٠.

In step٢, the same process is applied to L_2, and the result is the ٤-bit string L_3=١١٠١, which is short enough so no more compression steps are needed. After deleting all zero substrings in L_1 and L_2 we end up with the three short strings

L_3= ١١٠١. L_2 = ٠٠٠١ | ٠٠٠١ | ١٠٠٠, L_1 = ٠١٠٠ | ١٠٠٠ | ٠٠١٠,

The output stream consists of seven ٤-bit sub strings instead of the original ١٦ (A few more numbers are needed, to indicate how long each sub string is).

The decoder works differently (this is an asymmetric compression method). It starts with L_3 and considers each of its ١ bits a pointer to a sub string of L_2 and each of its ٠ bits a pointer to a sub string of all zero that's not stored in L_2. This way string L_2 can be reconstructed from L_3, and string L_1, in

turn, from L_2. Figure (٢-٦) illustrates this process. The substrings shown in square brackets are the ones not contained in the compressed stream.

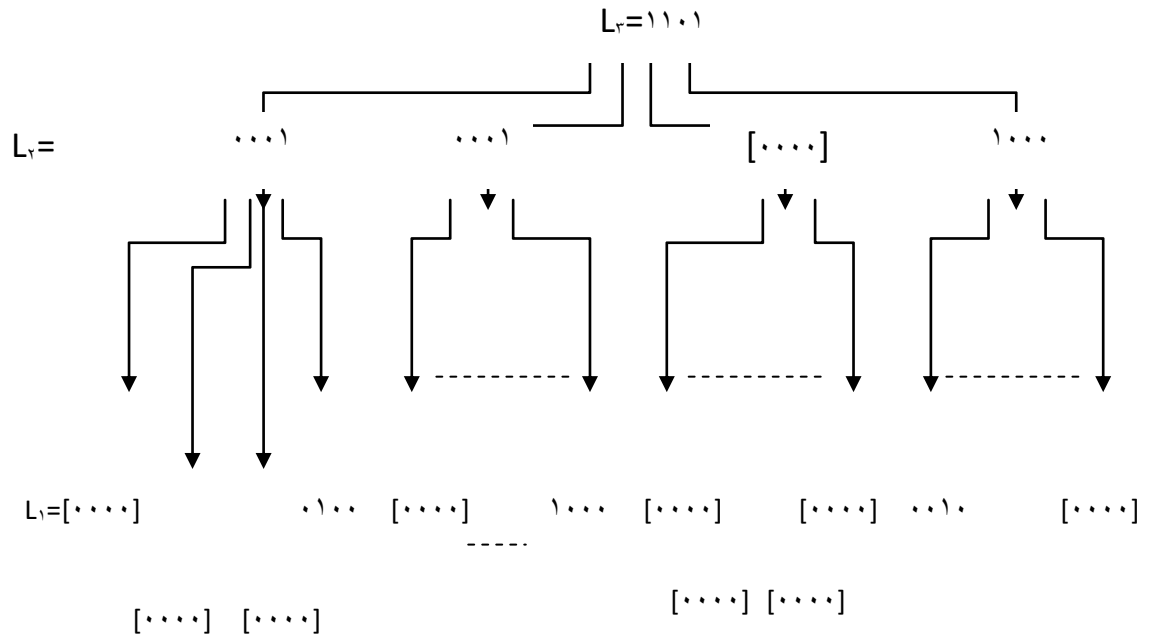


Figure (٢-٦): Reconstructing Process in Oring Bits.

٢.٢.٦ Shannon Method [٨]

This was the first method developed for finding good, variable size code. We start with a set of n symbols with known probabilities (or frequencies) of occurrence. The symbols are parsed arranged in descending order of their probabilities. The set of symbols is then divided into two subsets that have the same (or almost the same) probabilities. All symbols in one subset get assigned codes that start with a ٠, while the codes of the symbols in the other subset start a ١. Each subset is then recursively divided into two, and the second bit of all

the code is determined in a similar way. When a subset contains just two symbols, their codes are distinguished by adding one more bit to each. The process continues until no more subsets remain.

The following example is illustrated in the figure(٢-٧):

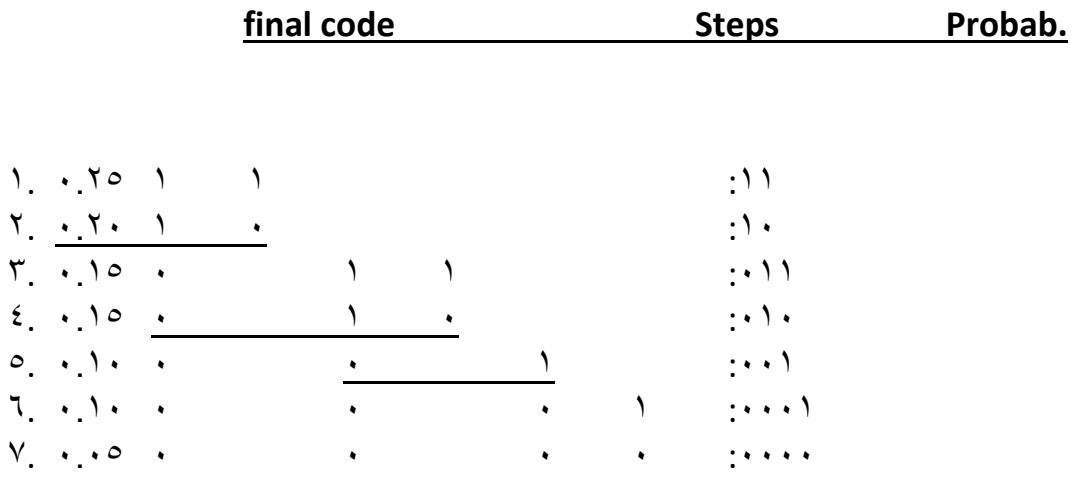


Figure (٢-٧) Shannon Coding

٢.٢.٧ *Huffman Method [٨, ٢٩, ١٨, ٢٦]*

In ١٩٥١, David Huffman and his MIT information theory classmates were given the choice of a term paper or a final exam. The professor, Robert M. Fano, assigned a term paper on the problem of finding the most efficient binary code. Huffman, unable to prove any codes were the most efficient, was about to give up and start studying for the final when he hit upon the idea of using a frequency-sorted binary tree, and quickly proved this method the most efficient.

The algorithm of Huffman is:-

A fast way to create a Huffman tree is to use the heap data structure, which keeps the nodes in partially sorted order according to a predetermined criterion. In this case, the node with the lowest weight is always kept at the root of the heap for easy access.

Creating the tree:

١. Start with as many leaves as there are symbols.
٢. Push all leaf nodes into the heap.
٣. While there is more than one node in the heap:

-Remove two nodes with the lowest weight from the heap.

-Put the two nodes into the tree, noting their location.

-If parent links are used, set the children of any internal nodes to point at their parents.

-Create a new internal node, using the two nodes as children and their combined weight as the weight.

-Push the new node into the heap.

٤. The remaining node is the root node.

Note it may be beneficial to generate codes with minimum length variance, since in the worst case when several long codewords have to be transmitted in a row it may lead to unwanted side effects (for example if we use a buffer and constant rate transmitter, it increases the minimum buffer size). To reduce variance every newly generated node must be favoured among same

weight nodes and placed as high as possible. This will balance the length, keeping same average rate.

٢.٢.٧.١ *Main Properties For Huffman Method[١٥]*

The frequencies used can be generic ones for the application domains that are based on average experience, or they can be the actual frequencies found in the text being compressed. (This variation requires that a frequency table or other hints as to the encoding must be stored with the compressed text; implementations employ various tricks to store these tables efficiently).

Huffman coding is optimal when the probability of each input symbol is a negative power of two. Prefix-free codes tend to have slight inefficiency on small alphabets, where probabilities often fall between these optimal points. Expanding the alphabet size by coalescing multiple symbols into "words" before Huffman coding can help a bit. The worst case for Huffman coding can happen when the probability of a symbol exceeds $٢^{-١}$ making the upper limit of inefficiency unbounded. To prevent this, run-length encoding can be used to preprocess the symbols.

٢.٢.٧.٢ Minimum Variance Huffman Codes.

The Huffman coding algorithm has some flexibility when two equal frequencies are found. The choice made in such situations will change the final code including possibly the code length of each message. Since all Huffman codes are optimal, however, it cannot change the average length [١٣].

For example, consider the following message probabilities, and codes in Table (٢-١):

Table (٢-١) Huffman Coding

Symbol	Probability	Code ١	Code ٢
A	٠.٢	٠١	١٠
B	٠.٤	١	٠٠
C	٠.٢	٠٠٠	١١
D	٠.١	٠٠١٠	٠١٠
E	٠.١	٠٠١١	٠١١

Code ١ : ٠.٤*١ + ٠.٢*٢ + ٠.٢*٣ + ٠.١*٤ + ٠.١*٤ = ٢.٢

Code ٢ : ٠.٤*٢ + ٠.٢*٢ + ٠.٢*٢ + ٠.١*٣ + ٠.١*٣ = ٢.٢

Both codes produce an average of ٢.٢ bits per symbol, even though the lengths are quite different in the two codes. Given this choice, is there any reason to pick one code over the other?

For some applications it can be helpful to reduce the variance in the code length. The variance is defined as:

$$\text{Variance} = \sum_{i=1}^{n} p_i (a_i - A)^2 \quad \ldots(2.2)$$

where,

p: probability

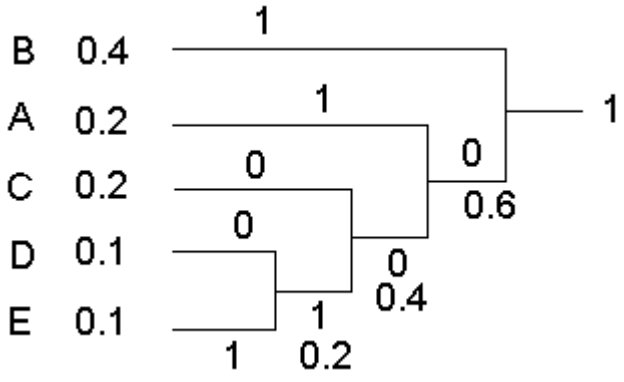
a_i: number of bits(length code word)

A: average size

$$0.4\ (1-2.2)^2 + 0.2\ (2-2.2)^2 + 0.2\ (3-2.2)^2 + 0.1\ (4-2.2)^2 + 0.1\ (4-2.2)^2 = 1.36$$

$$0.4\ (2-2.2)^2 + 0.2\ (2-2.2)^2 + 0.2\ (2-2.2)^2 + 0.1\ (3-2.2)^2 + 0.1\ (3-2.2)^2 = 0.16$$

With lower variance it can be easier to maintain a constant character transmission rate, or reduce the size of buffers. In the above example, code 1 clearly has a much higher variance than code 2 and it is illustrated in Figure (2-8) and Figure (2-9) respectively.



Figure(٢-٨) : Huffman Tree (code ١).

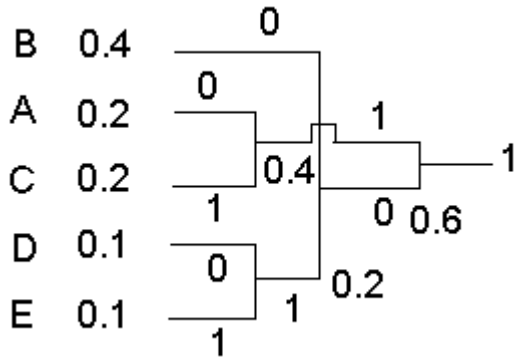


Figure (٢-٩) Huffman Tree(code ٢).

It turns out that a simple modification to the Huffman algorithm can be used to generate a code that has minimum variance. In particular when choosing the two nodes to merge and there is a choice based on weight, always pick the node that was created earliest in the algorithm. Leaf nodes are assumed to be created before all internal nodes. In the example above, after D and E are joined, the pair will have the same probability as C and A (٠.٢), but it was created afterwards, so we join C and A. Similarly we select B instead of the join AC to join with DE since it was created earlier. This will give code ٢ above, and the corresponding Huffman tree in Figure (٢-٩).

٢.٢.٨ *Adaptive Huffman Coding[٨]*

Traditional Huffman Coding uses a static dictionary, which means each item in the file is encoded the same way. Typical implementations send the encoding table to the decoder as part of the encoded file.

Adaptive Huffman coding modifies the table as characters are encoded, which allows the encoder to adapt to changing conditions in the input data. Adaptive decoders do not need a copy of the table when decoding, they start with a fixed decoding table and update the table as characters are read in.

٣.١ Introduction [٣٦,١٠]

A **genetic algorithm** (**GA**) is a heuristic used to find approximate solutions for difficult to solve problems through application of the principles of evolutionary biology to computer science. Genetic algorithms use biologically-derived techniques such as inheritance, mutation, natural selection, and recombination (or crossover). Genetic algorithms are a particular class of evolutionary algorithms.

Genetic algorithms are typically implemented as a computer simulation in which a population of abstract representations (called **chromosomes**) of candidate solutions (called **individuals**) to an optimization problem evolving toward better solutions. Traditionally, solutions are represented in binary as strings of ٠s and ١s, but different encodings are also possible. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population

(based on their fitness), modified (mutated or recombined) to form a new population, which becomes current in the next iteration of the algorithm.

٣.٢ Basic Description [٣٥]

Genetic algorithms are inspired by Darwin's theory about evolution.

Solution to a problem solved by genetic algorithms is evolved.

Algorithm is started with a **set of solutions** (represented by **chromosomes**) called **population**. Solutions from one population are taken and used to form a new population. This is motivated by a hope that the new population will be better than the old one. Solutions which are selected to form new solutions (**offspring**) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.

This is repeated until some condition (for example a number of populations or improvement of the best solution) is satisfied.

٣.٣ Chromosome [٢٠]

All living organisms consist of cells. In each cell there is the same set of **chromosomes**. Chromosomes are strings of **DNA** and serve as a model for the whole organism. A chromosome consists of **genes**, blocks of DNA. Each gene encodes a particular protein. Basically it can be said, that each gene encodes a **trait**, for example color of eyes. Possible settings for a trait (e.g. blue, brown) are called **alleles**. Each gene has its own position in the chromosome. This position is called **locus**.

A complete set of genetic material (all chromosomes) is called **genome**.

A particular set of genes in genome is called **genotype**. The genotype is with later development after birth base for the organism's **phenotype**, its physical and mental characteristics, such as eye color, intelligence etc.

٣.٤ Outline of the Basic Genetic Algorithm [١,٢٠]

١. **Start** Generate random population of *n* chromosomes (suitable solutions for the problem).

٢. **Fitness** Evaluate the fitness *f(x)* of each chromosome *x* in the population

٣. **New population** Create a new population by repeating following steps until the new population is complete:

   a. **Selection** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).

   b. **Crossover** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, an offspring is an exact copy of parents.

   c. **Mutation** With a mutation probability mutate a new offspring at each locus (position in chromosome).

   d. **Accepting** Place a new offspring in a new population .

٤. **Replace** Use new generated population for a further run of algorithm

٥. **Test** If the end condition is satisfied, **stop**, and return the best solution in current population.

٦. **Loop** Go to step **٢.**

As you can see, the outline of Basic GA is very general. There are many things that can be implemented differently in various problems.

The first question is how to create chromosomes, what type of encoding is chosen. This is connected with crossover and mutation which are the two basic operators of GA.

The next questions is how to select parents for crossover. This can be done in many ways, but the main idea is to select the better parents (in the hope that the better parents will produce better offspring). In this case, there is a big chance the best chromosome is lost. That is true, here is where **elitism** is used.

٣.٤.١ Initialization

The Algorithm is started with a set of solutions (represented by chromosomes) called population.

Each chromosome represents a possible solution to the problem. The most-used way (though not the only way) of encoding chromosomes is a binary string [١].

٣.٤.٢ Calculation of Fitness Function [١]

An evaluation function, called fitness function, needs to be defined for a problem to be solved in order to evaluate chromosomes. The fitness of a chromosome defines how well it is suited for the problem; hence it determines its likelihood for survival from one generation to the next. A chromosome with a high fitness value is likely to be a good solution to the problem.

٣.٤.٣ Selection of the Best Individuals:

As you already know from the GA outline, chromosomes are selected from the population to be parents to crossover. The problem is how to select

these chromosomes. According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others [١].

Some of them will be described in this section:

٣.٤.٣.١ Roulette Wheel Selection [٢٠]

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a **roulette wheel** where all chromosomes are placed in the population, every one has its big place accordingly to its fitness function.

Then a marble is thrown there and selects the chromosome. Chromosome with bigger fitness will be selected more times.

This can be simulated by the following algorithm.

١. Find the fitness f_i of each member of the population.

٢. Find the sum of the fitnesses:

$$S = \sum_{i=1}^{n} f_i$$

٣. Scale each fitness with S (convert to a probability):

$$p_i = f_i / S.$$

٤. Find the cumulative probability for each individual:

$$c_i = \sum^{i} p_k \quad \text{for} \quad i = 1, \ldots N$$

k=١

٥. Choose a random number r uniformly from [٠, ١] and select member m_i if r is such that $c_{i-١} < r \leq c_i$. (Select $m_١$ if $r < c_١$.) Do this N times.

٣.٤.٣.٢ Rank Selection [٢٠]

The previous selection will have problems when the fitnesses differs very much. For example, if the best chromosome fitness is ٩٠٪ of all the roulette wheel, then the other chromosomes will have very few chances to be selected.

Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness ١, second worst ٢ etc. and the best will have fitness ***N*** (number of chromosomes in population).

After this all the chromosomes have a chance to be selected. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.

٣.٤.٣.٣ Steady-State Selection [٢٠,١٦]

This is not a particular method of selecting parents. The main idea of this selection is that a big part of chromosomes should survive to the next generation.

GA then works in a following way. In every generation a few chromosomes (good - with high fitness) are selected for creating a new offspring. Then some (bad - with low fitness) chromosomes are removed and the new offspring is placed in their place. The rest of population survives to new generation.

٣.٤.٣.٤ Tournament Selection [٢]

This method of selection organizes the number of copies inclusively by giving the individuals with the least fitness a chance to be selected. There are many forms to perform this kind of selection. The simplest and more common is binary tournament selection that is used in (ssGA). Where two different individuals are selected randomly, the best wins and is added to the crossover pool. In the case the two fitness are equal, one of them is selected randomly. This operation is repeated for selecting the other individual that will share in the crossover operation.

١. Select k individuals (at random) from the current population.

٢. Choose the best of these individuals to be placed in the new population.

- If more than one of the selected individuals has the same fitness choose the winner at random.

٣. Repeat this N times until the new population has been formed.

Notes:

- k = ٢ is often used.
- k = N would result in a new population consisting solely of the best individual(s).

The algorithm above assumes sampling with replacement [١١].

٣.٤.٤ Crossover [٣٣]

One of the unique aspects of the work involving genetic algorithms (GAs) is the important role that crossover plays. Crossover operators produce two new chromosomes by exchanging information of the selected chromosomes. Good individuals will probably be selected several times in a generation, poor ones may not be at all. This operator is the most essential in genetic algorithms.

٣.٤.٤.١ Traditional Crossover

After we have decided what encoding we will use, we can make a step to crossover. Crossover selects genes from parent chromosomes and creates a new offspring. The simplest way how to do this is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after a crossover point copy from the second parent. In other words ,Crossover selects random genes from parent chromosomes and creates a new offspring. The chromosomes of the two parents are split into two (equal or unequal) halves each. Both chromosomes are cut similarly. The halves are interchanged and combined to form the child chromosome [١,٣٣]. The following illustrates some types of this kind of crossover.

١- One-Point Crossover (١X) [٢٣,٢٧]

Suppose we are using a bit-string representation and that one of the chosen pairs is ١٠١٠١١٠١٠٠١١ and ١١١١٠٠١١٠١١٠ (each of length ١٢).

- Suppose a number at random between ١ and k − ١ has been chosen (here ١١).
- Suppose it is ٨. This is the crossover position (marked by the bar).

١٠١٠١١٠١|٠٠١١

١١١١٠٠١١|٠١١٠

- All bits from position ٨ to the end of the string are swapped. The new strings will be (bar left in to help clarify things).

١٠١٠١١٠١|٠١١٠

١١١١٠٠١١|٠٠١١

٢- Two-Point Crossover (٢X) [٢٣,٧]

Suppose we are using a bit-string representation and that one of the chosen pairs is ١٠١٠١١٠١٠٠١١ and ١١١١٠٠١١٠١١٠ (each of length ١٢).

- Choose two numbers at random between ١ and k.

- Suppose they are ٣ and ٩. These are the crossover positions (indicated by the bars)

١٠١|٠١١٠١٠|٠١١

١١١|١٠٠١١٠|١١٠

- Swap the bits between the markers.

١٠١|١٠٠١١٠|٠١١

١١١|٠١١٠١٠|١١٠

(The idea can be generalized to n-point crossover.)

٣.٤.٤.٢ Permutation Crossover

The operators of permutation are:

a- Partially Mapped Crossover (PMX)[٢٢].

PMX is implemented as follows:

Choose a random cut point and consider the segments following the cut point in both parents as a partial mapping of the cells to be exchanged in the first parent to generate the offspring. Take corresponding cells from the segments of both parents, locate both these cells in the first parent, and exchange them. Repeat this process for all cells in the same location in the second parent will define which cells in the first parent have to be exchanged to generate the offspring. An example is shown in Figure (٣-١), where the first parent is **abcdefghij**, the second parent is eciadhjbfg, and the point cut is at the seventh cell.

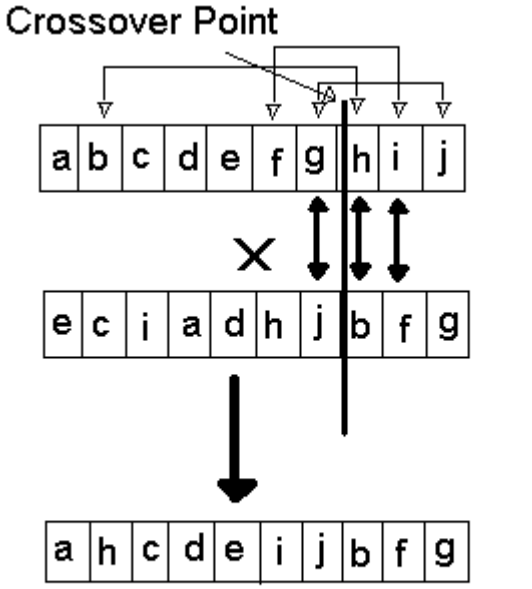
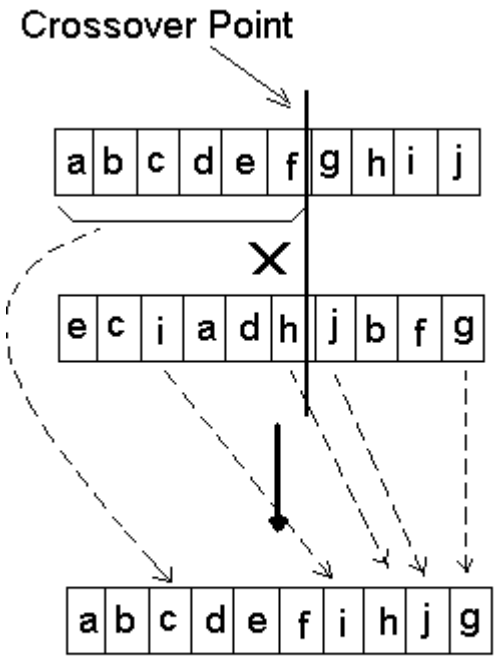


Figure (3-1) PMX Crossover

b- Order Crossover (OX)[٢٢].

This kind of crossover is implemented as follows:

Choose a cut point at random. Copy the array segment to the left of the cut point from one parent to the offspring. Fill the remaining (right) portion of the offspring array by going through the second parent, from the beginning to the end and taking those elements which were left out, in order. An example is shown in Figure (٣-٢). This operator conveys a sub-placement from the first parent without any changes. And then, to resolve conflicts, compress the second parent by eliminating the cells conveyed by the first parent and shifting the rest of the cells to the left without changing their order. It then copies this compressed second parent into the remaining part of the offspring array [٢٢].



Figure(3-2) Order Crossover

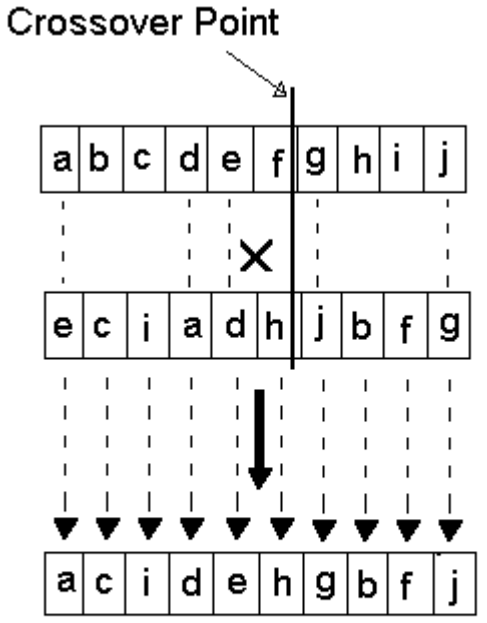
c- Cycle Crossover (CX)[٢٢].

Cycle crossover is an attempt to eliminate the cell conflicts that normally arise in crossover operators. In the offspring cells generated by cycle crossover, every cell is in the same location as one of the parents. The basic idea behind cycle crossover is to avoid cell conflicts by finding non-overlapping sets of cells to pass from the two parents. However, such non-overlapping sets of cells are not contiguous. This kind of crossover is implemented as follows:

Start with the cell in location ١ (or any other reference point), and copy it to location ١ of the offspring . Now consider what will happen to the cell in location ١ of parent ٢. The offspring cannot inherit this cell from parent ٢, since location ١ in the offspring has been filled. Therefore, this cell must be obtained from parent ١. Suppose this cell is located in parent ١ at location X. Then it is passed to the offspring at location X. But then the cell at location X in parent ٢ cannot be passed to the offspring, so that cell is also obtained from parent ١. This process continues until we complete a cycle and reach a cell that has already been passed. Then we choose a cell from parent ٢ to pass the

offspring and go through another cycle, passing cells from parent ٢ to the offspring. Thus, in alternate cycles, the offspring inherits cells from alternate parents, and the cells are placed in the same locations as they were in the parents from which they were inherited.

An example is given in Figure (٣-٣). In this example, we start by passing cell **a** from parent ١ to the offspring. Since **e** is located in the same position in parent ٢, it cannot be passed from parent ٢. Hence, **e** is also passed from parent ١. **d** is located in parent ٢ in the same position as **e** in parent ١. Hence **d** must also be passed from parent ١, and so on until we reach cell **a** again. This completes the cycle. The next cycle is taken from parent ٢ and consists of the cells **c**, **b**, **h**, **f**, and **i**. The third cycle is again from parent ١ and consists of cells **g** and **j** .



Figure(3-3) Cycle crossover

٣.٥ Mutation [٢٠,١]

After a crossover is performed, the resulting solution might fall into a local optimum. Mutation changes randomly the new offspring. (In Figure(٣-٤), bits are randomly toggled in case of binary strings).

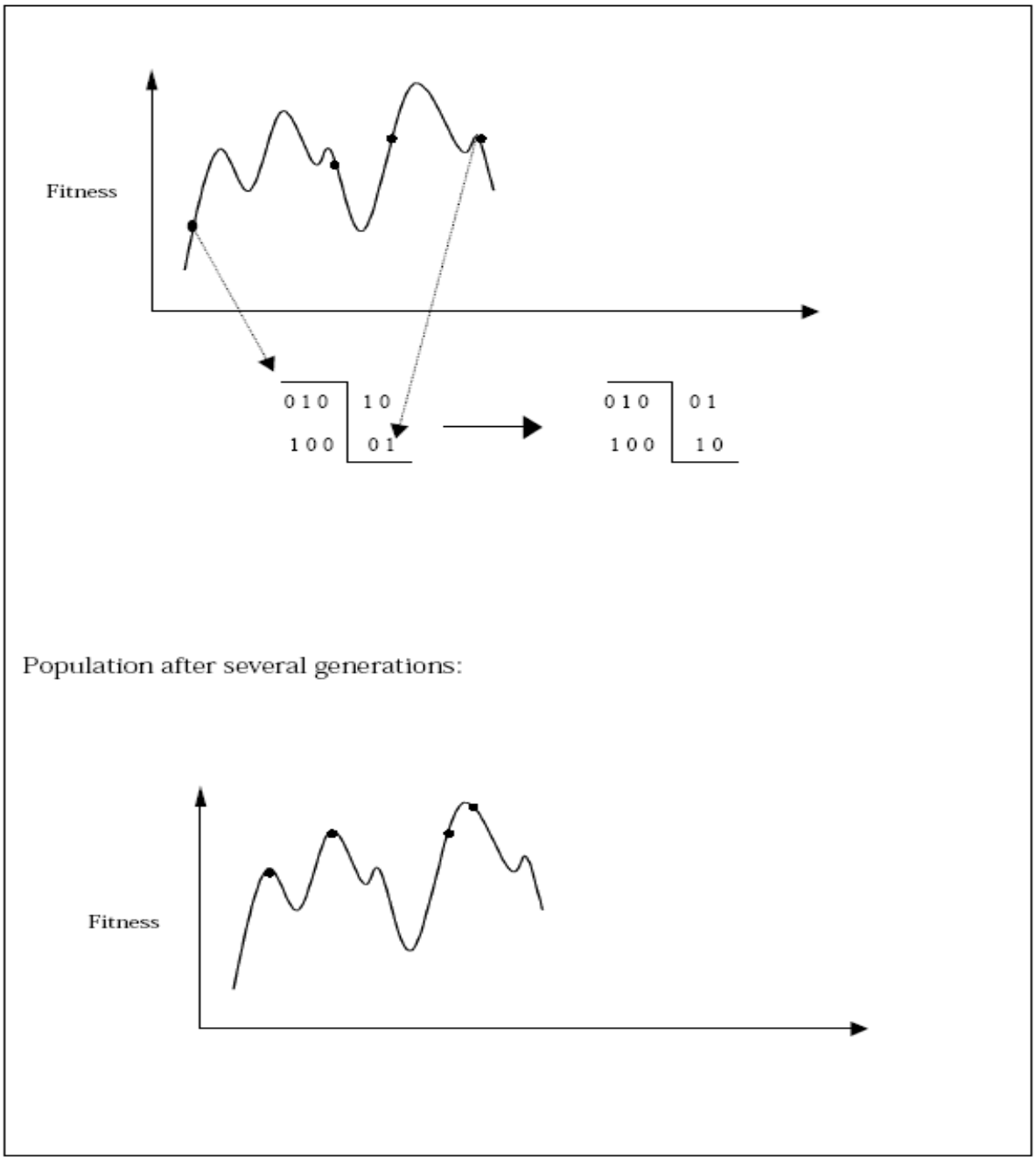


Figure (٣-٤) Mutation Effect

For binary encoding we can switch a few randomly chosen bits from ١ to ٠ or from ٠ to ١. Mutation can then be as follows:

Original offspring ١	١١٠١١١١٠٠٠٠١١١١٠
Original offspring ٢	١١٠١١٠٠١٠٠١١٠١١٠
Mutated offspring ١	١١٠٠١١١٠٠٠٠١١١١٠
Mutated offspring ٢	١١٠١١٠١١٠٠١١٠١١٠

Figure (٣-٥) Mutation Example

The mutation depends on the encoding as well as the crossover. For example when we are encoding permutations, mutation could be exchanging two genes.

Mutation changes a gene in the chromosome with small probability Pm. Mutation is a mechanism for restoring lost and unexplored genetic material into the population and searching regions of the allele space not generated by selection and crossover.

It can be used to prevent premature conversion. This increases the diversity of genetic material.

There are many methods of mutation:

١- Flips the value of a gene from ١ to ٠ or vice versa and as in the previous example.

٢- Change the value of a gene.

٣- Swap the value of two genes.

٣.٦ Crossover and Mutation Probability

There are two basic parameters of GA - crossover probability and mutation probability. **Crossover probability** says how often crossover will be

performed. If there is no crossover, an offspring is exact copy of parents. If there is a crossover, the offspring is made from parts of parents' chromosome. If crossover probability is ١٠٠٪, then every offspring is made by crossover. If it is ٠٪, a whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!). Crossover is made in the hope that new chromosomes will have good parts of old chromosomes and the new chromosomes maybe better. However it is good to leave some part of population survive to the next generation.

Mutation probability says how often will parts of the chromosome be mutated. If there is no mutation, offspring is taken after crossover (or copy) without any change. If mutation is performed, part of the chromosome is changed. If mutation probability is ١٠٠٪, the whole chromosome is changed, if it is ٠٪, nothing is changed. Mutation is made to prevent falling GA into local extreme, but it should not occur very often, because then GA will in fact change to **random search**.

٣.٧ Inversion

The inversion operator takes a random segment in a solution string and inverts its end for end Figure (٣-٦). This operation is performed in such a way that it does not modify the solution represented by the string. Instead, it only modifies the representation of the solution. Thus, the symbols composing the string must have an interpretation independent of their position [٢٢].

Before Inversion:	٩ ١	٨ ١	٧ ٠	٦ ٠	٥ ٠	٤ ٠	٣ ١	٢ ١	١ ٠	٠ ٠
After										

er	٩	٨	٧	٢	٣	٤	٥	٦	١	٠
Inversion	١	١	٠	١	١	٠	٠	٠	٠	٠

Figure (٣-٦) Inversion

Inversion changes the sequence of genes randomly, in the hope of discovering a sequence of linked genes placed close to each other. The inversion operator greatly improves the efficiency of the crossover operator.

٣.٨ Replacement

Replacement deletes old individuals with least fitness and replaces them with new individuals of better fitness; the following are two common methods for replacement [٥]:

١- Binary tournament Replacement[٥]

This replacement method is identical to the binary tournament selection illustrated in section (٣.٤.٣.٤), where the new offspring that is evaluated from both the crossover and mutation operation is replaced with worst individual selected from the two randomly selected individuals from the population. The following mathematical form illustrates this type of replacement:

$$\text{Replace}_n = \begin{cases} Ind_i & \text{if } F(ind_i) < F(ind_j) \\ Ind_j & \text{otherwise} \end{cases} \quad \ldots.(٣.١)$$

For $n=\{١,٢\}$, random number $i, j \in \{١,٢,\ldots.,N_{pop}\}$ $i \neq j$.

$Replace_n$: individual n wanted replace .

$F(ind_i)$: the fitness of individual $_i$.

$F(ind_j)$: fitness of individual $_j$.

This type of replacement methods always keeps the best individuals and by elitism size equal to one at least. To increase the probability of keeping the best individuals in the population there is another method which can be implemented as illustrated in the following replacement method.

٢- Triple Tournament Replacement [٥]

This type is identical to the previous type of replacement except replacing the new offspring instead of the worst individual selected from three randomly selected individuals from the population. The following mathematical form illustrates that:

$$Replace_n = \begin{cases} ind_i & \text{if } F(ind_i) < F(ind_j) \text{ and } F(ind_i) < F(ind_k) \\ ind_j & \text{if } F(ind_j) < F(ind_i) \text{ and } F(ind_j) < F(ind_k) \\ ind_k & \text{otherwise} \end{cases} \quad ..(٣.٢)$$

For n={١,٢} ,and random number i ,j,k ∈ {١,٢,….,N_{pop}} i ≠ k, j≠k

This type of replacement gives a higher probability from the previous type from keeping the best individuals, where it is produced implicit a elitism size equal to two or less (the best two individuals are never replaced).

٣- Other Replacement Methods

In addition to the two previous methods there are other replacement methods:

١- Worst Replacement: Replacing worst offspring with the worst individual in the population [١٩].

٢- Random Replacement: Replacing the new offspring with the individual that is randomly selected from the population [١٩].

٣- In addition to that a strategy replacement may be used that takes in consider the fitness and individual's similarity to the population in order to choose the desired individual to be deleted which has a low fitness and is relatively similar to the new child [٦].

٣.٩ Encoding [٢١]

Encoding of chromosomes is one of the problems, when you are starting to solve a problem with GA. Encoding deeply depends on the problem. This section will introduce some encoding, which has been used with some success.

٣.٩.١ Binary Encoding

Binary encoding is the most common, mainly the because first works about GA used this type of encoding.

In **binary encoding** every chromosome is a string of **bits**, ٠ or ١.

The example of chromosomes with binary encoding is the following:

Chromosome A	١٠١١٠٠١٠١١٠٠١٠١٠١١١٠٠١٠١
Chromosome B	١١١١١١١٠٠٠٠٠١١٠٠٠٠٠١١١١١

Binary encoding gives many possible chromosomes even with a small number of alleles. On the other hand, this encoding is often not natural for many problems and sometimes corrections must be made after crossover and/or mutation.

٣.٩.٢ Permutation Encoding [٢١]

Permutation encoding can be used in ordering problems such as the traveling salesman problem or the task ordering problem.

In **permutation encoding**, every chromosome is a string of numbers, which *represents number in a **sequence***.

Example of chromosomes with permutation encoding is:-

Chromosome A	١ ٥ ٣ ٢ ٦ ٤ ٧ ٩ ٨
Chromosome B	٨ ٥ ٦ ٧ ٢ ٣ ١ ٤ ٩

Permutation encoding is only useful for ordering problems. Even for this problems for some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e. have real sequence in it).

٣.٩.٣ Value Encoding [٢١]

Direct value encoding can be used in problems, where some complicated values, such as real numbers, are used. The use of binary encoding for this type of problems would be very difficult.

In **value encoding**, every chromosome is a string of some values. Values can be anything connected to the problem, form numbers, real numbers or chars to some complicated objects.

Example of chromosomes with value encoding is:

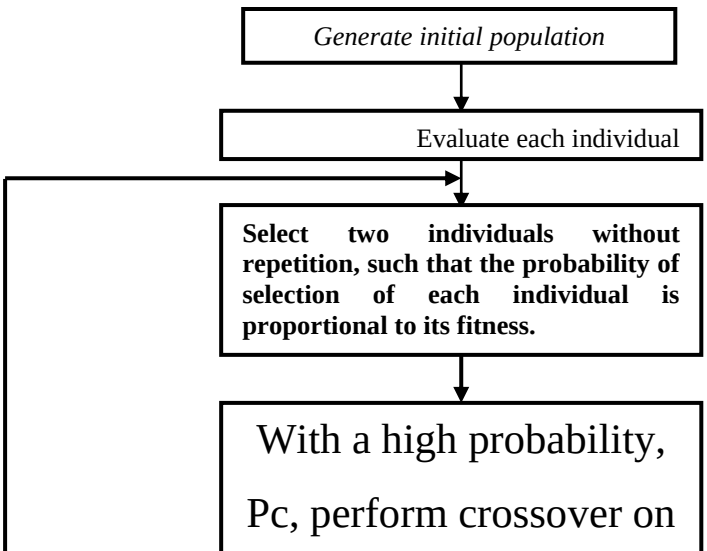
Chromosome A	١.٢٣٢٤ ٥.٣٢٤٣ ٠.٤٥٥٦ ٢.٣٢٩٣ ٢.٤٥٤٥
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Value encoding is very good for some special problems. On the other hand, this encoding is often necessary to develop some new crossover and mutation specific for the problem.

٣.١٠ Steady State Genetic Algorithm (ssGA).

In a GA having overlapping generations, only fraction of the individuals are replaced in each generation. The steady state algorithm is illustrated in Figure (٣-٧). In each generation, two different individuals are

selected as parents, based on their fitness. Crossover is performed with a high probability, Pc, to form offspring. The offspring are mutated with a low probability,Pm and inverted with probability PI, if necessary. A duplicate check may follow, in which the offspring are rejected without any evaluation if they are duplicates of some chromosomes already in the population. The offspring that survive the duplicate check are evaluated and are introduced into the population only if they are better than the current worst member of the population, in which case the offspring replaces the worst member. This completes the generation. In the steady state GA, the generation gap is minimal, since only two offspring are produced in each generation [٢٢,١١].



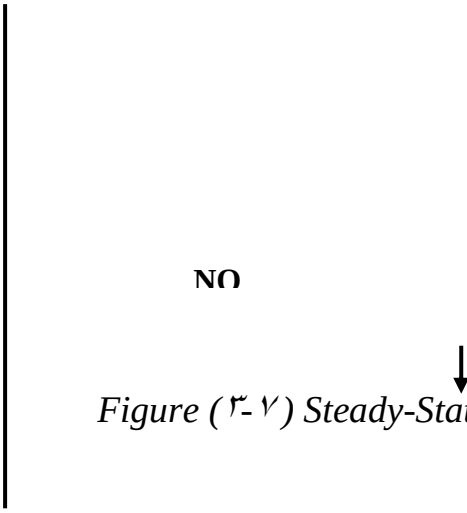


Figure (٣-٧) Steady-State Genetic Flowchart

٣.١٠.١ Why Steady-State GA (ssGA) Used ?

Simple genetic algorithm SGA replaces the entire parent population with the children; one tactic that is often added to a selection strategy is elitism. In elitism the fittest individuals pass unchanged from the parent population to the children one. This is to ensure that the fittest genes are not lost to the next generation.

Steady state GA (ssGA) replaces few individuals, while a standard GA aims at finding a single Global optimum, the application of the concept of the Niching methods (subpopulation) provide a set of solutions rather than only one solution [٢٢].

٣.١١ Population Size [٢٠]

There are also some other parameters of GA. An important parameter is also population size. **Population size** says how many chromosomes are in population (in one generation). If there are too few chromosomes, GAs have a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to increase the population size, because it does not make solving the problem faster.

٣.١٢ Elitism [٢٠]

When creating a new population by crossover and mutation, we have a big chance, that we will loose the best chromosome. Elitism is the name of method, which first copies the best chromosome (or a few best chromosomes) to a new population. The rest is done in a classical way. Elitism can very rapidly increase performance of GA, that at least one best solution is copied without changes to a new population, so the best solution found can survive to end of run.

٣.١٣ Stopping Criteria[٣١]

There are many criteria for stopping the GA process as:

-After a fixed number of generation.

-When known optimal fitness is reached.

-When GA can do no better when all individuals are identical.

٣.١٤ Applications of GA

Genetic algorithms have been used for difficult problems (such as NP-hard problems), for machine learning and also for evolving simple programs.

They have also been used for some art, for evolving pictures and music. Advantage of GAs is in their parallelism. GA is traveling in a search space with more individuals (and with genotype rather than phenotype) so they are less likely to get stuck in a local extreme like some other methods.

They are also easy to implement. Once you have some GA, you just have to write a new chromosome (just one object) to solve another problem. With the same encoding you just change the fitness function and that is all. On the other hand, choosing encoding and fitness function can be difficult. Disadvantage of GAs is in their computational time. They can be slower than some other methods. But with modren computers it is not so big problem.

Structure of proposed system

٤.١ Introduction

In this chapter, Figure (٤-١) illustrates the block diagram for the proposed system and its algorithms, and it has reviewed the system mechanism in compression based on some examples, where it illustrates the compression operation through building the trees and codes and applying them .

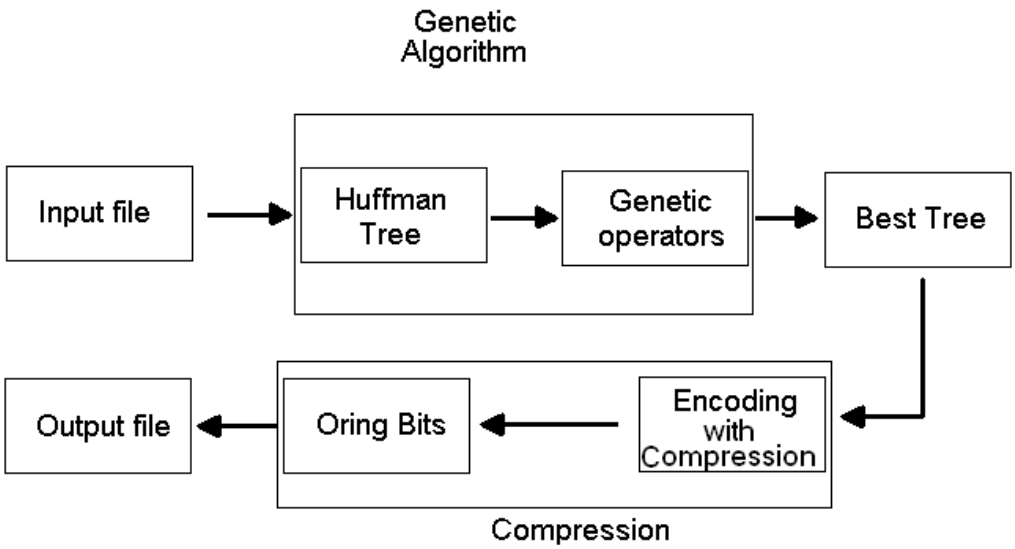
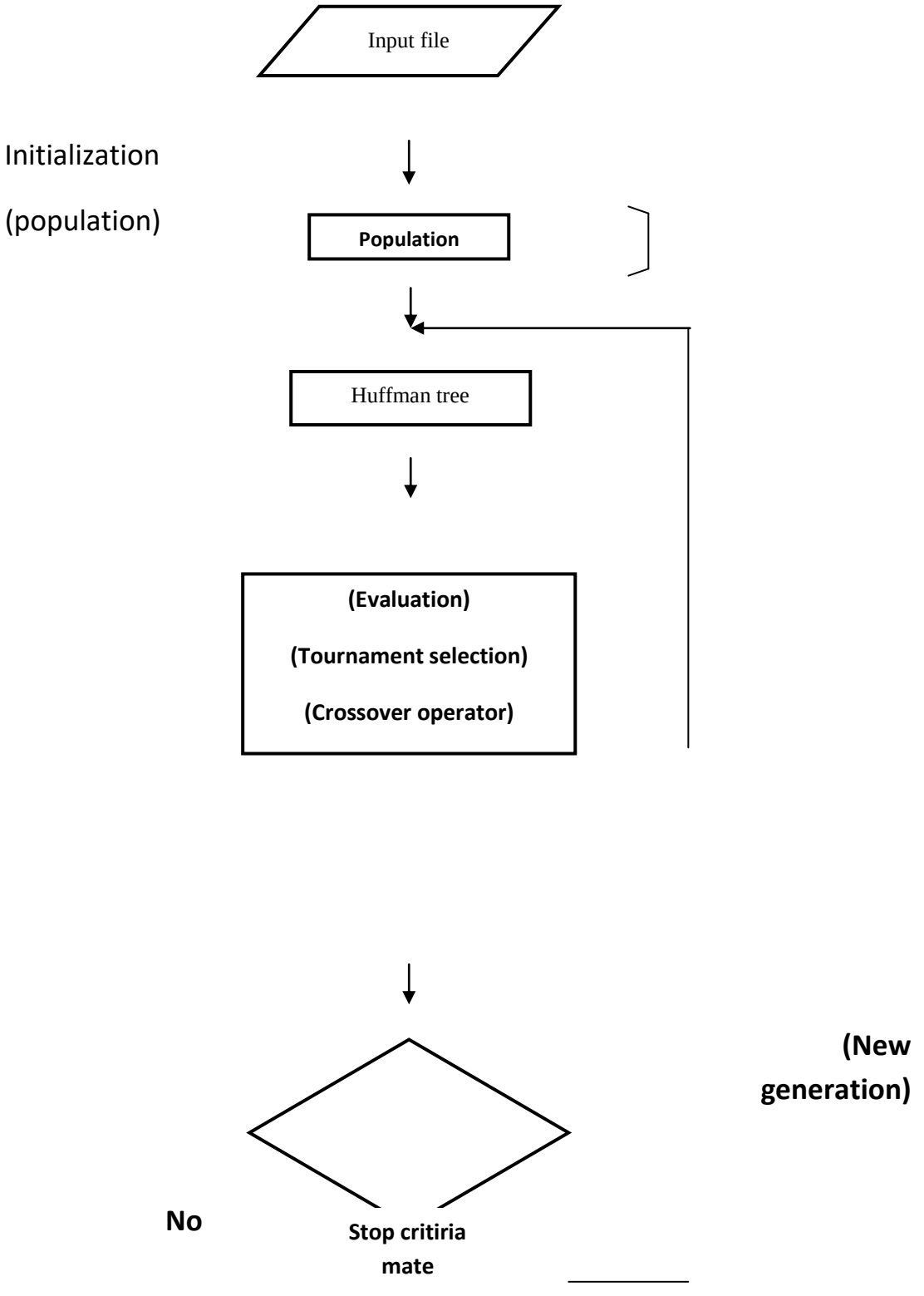


Figure (4-1) Block Daigram of Proposed System

The following flowchart in Figure (٤-٢) illustrates the direction of the system's work:


Input file
Initialization
(population)
Population
Huffman tree
(Evaluation)
(Tournament selection)
(Crossover operator)
(New generation)
No
Stop critiria mate

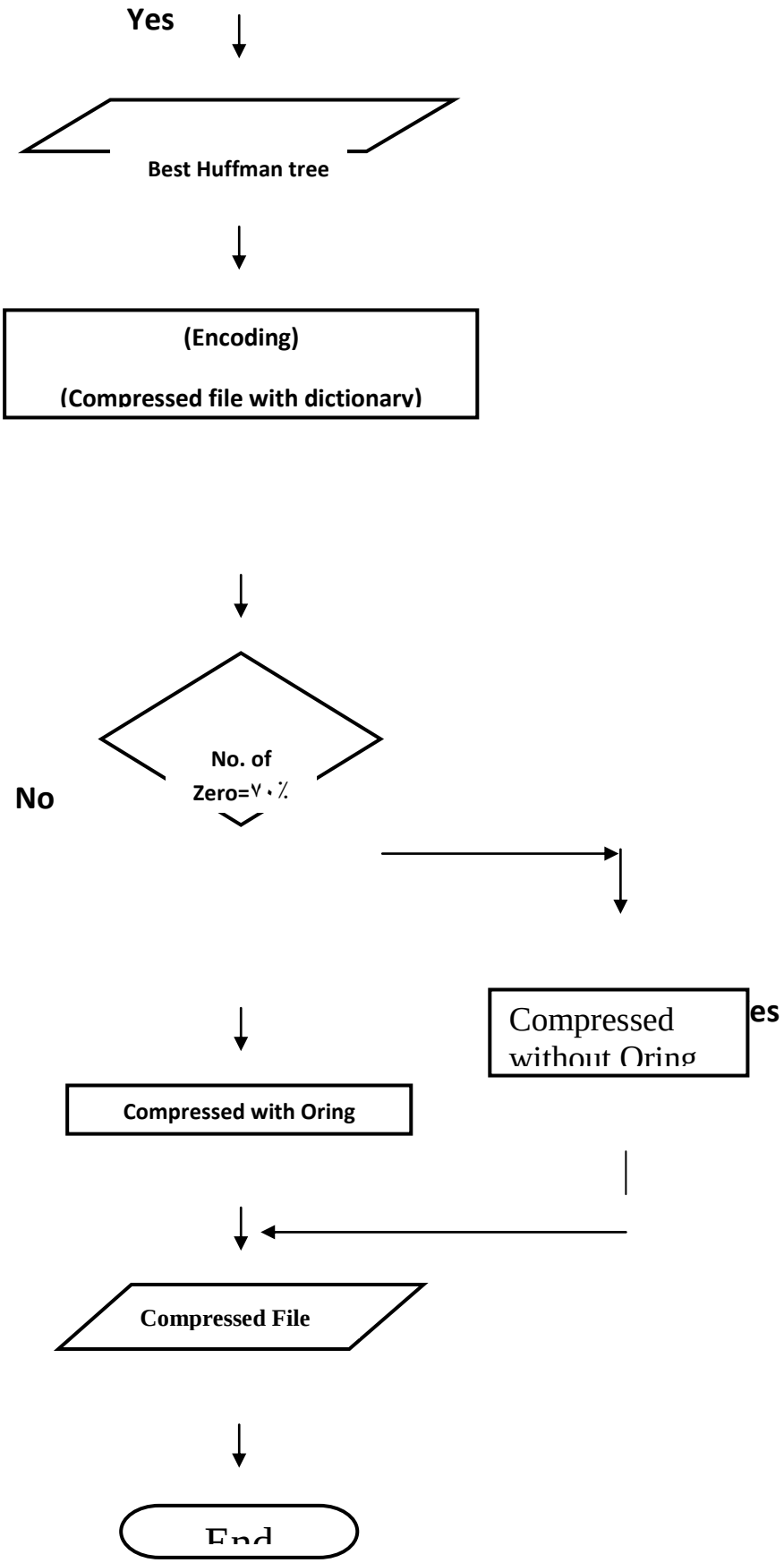


Figure (٤-٢)The General Flowchart for the Proposed System

٤.٢ *Creating the Individual & the Initial Population*

First, the text is wanted to be compressed and find the best Huffman tree for it is inputted from the notepad as a file. It has used text files which are read from the proposed system depending on the text; the initial population is created, where an individual is produced and from this individual (chromosome) the other individuals of the population are found randomly.

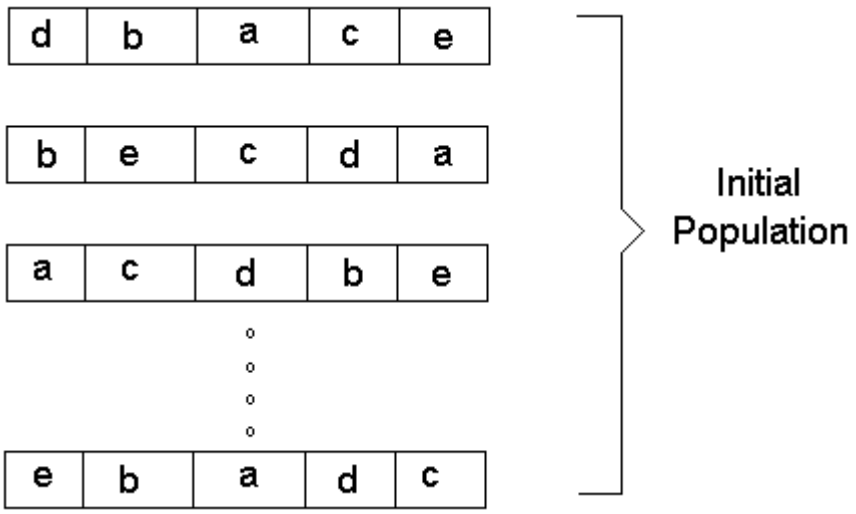
For example:

Text: abccddeeee

Produces the individual (chromosome)

a	b	c	d	e

:And from this individual the initial population is produced randomly as follows



The chromosome (individual) length depends on the variety in the text of the inputted characters. The frequency for each character is computed and then the probability for each character is found.

For example:

, c= ٠.٢ , d= ٠.٢ , e= ٠.٤ , b= ٠.١ a= ٠.١

The followed algorithm used to produce the individuals (chromosomes) and creating the initial population is:

١- Determining individual length.
٢- Determining the population size.
٣- Reading the input file and determining the first individual in the population.
٤- Generating random function to exchange the Genes locations and probabilities in order to create the rest of the population.

٤.٣ *Building Huffman Tree*

After generating the initial population, the Huffman tree is built for each individual in the population depending on their probabilities. The codeword is determined from the tree for each gene, for example in Figure (٤-٣):

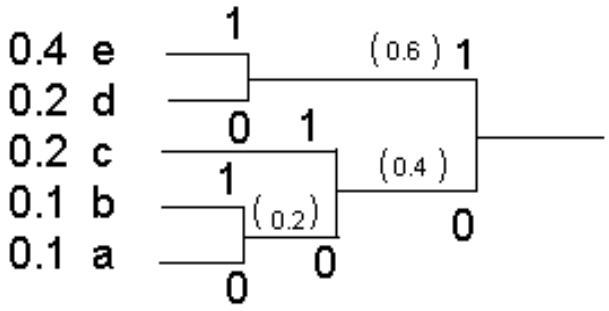


Figure (٤-٣) Huffman Codes

Where is:

e=١١ d=١٠ , c=٠١ , b=٠٠١ , a= ٠٠٠ ,

The followed algorithm used for building the Huffman tree is:

١- Computing the probability for each character in the text.

٢- Finding the two least probabilities according to the priority.

٣- Giving value '٠' for character with the biggest value from step ٢ and value '١' for the other, this is on the condition that the two characters have two different values. In the condition of having the same value, the upper (partial branch) gets value '٠' and the lower gets value '١'.

٤- Sum the two probabilities.

٥- The operation is repeated from step ٢ to step ٤ until the sum of two chosen probabilities is equal to '١'.

٦- Retrieve codeword for each character from right to left (probability).

Where the Huffman tree represents the dictionary that will associate the compressed file and is used in the decoding operation.

٤.٤ Evaluation

Every individual is evaluated by computing the fitness function. The fitness function here is the variance that depends on computing the average size for each individual.

$$A = \sum_{i=١}^{n} (P_i * a_i) \quad , \quad \text{.....(٤-١)}$$

$$\text{Variance} = \sum_{i=١}^{n} p_i (a_i - A)^{٢} \quad , \quad \text{.....(٤-٢)}$$

where,

p: probability

$a_{i:}$ number of bits(length code word)

A: average size

For example:
The average size for the previous example shown in Figure (٤-٣) is computed as follows:

٠.٤*٢ + ٠.٢*٢ + ٠.٢*٢ + ٠.١*٣ + ٠.١*٣ = ٢.٢

Then the variance is computed as follows:

$$٠.١٦٠ = ٠.١ (٣-٢.٢)^{٢} + ٠.١ (٣-٢.٢)^{٢} + ٠.٢ (٢-٢.٢)^{٢} + ٠.٢(٢-٢.٢)^{٢} + ٠.٤(٢-٢.٢)^{٢}$$

٤.٥ Tournament Selection

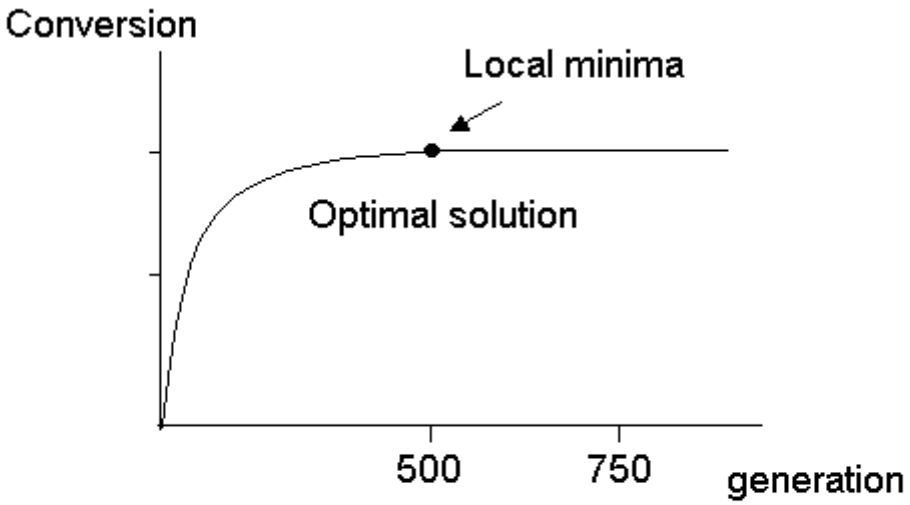
After the evaluation operation has been made, the two parents that will take place in the crossover operation are selected randomly.
Two randomly individuals are chosen to produce the subpopulation; the individual with the least variance is selected to be the first parent. The operation is repeated to get the second parent according to the same parameters above.

The followed algorithm used in the proposed system is:

١- Finding the subpopulation by selecting two individuals randomly.
٢- Selecting the individual with the best fitness and saving it; this is one of the two parents taking part in the crossover operation.
٣- The operation from step ١ to step ٢ is repeated to get the second selected parent.

Taking into consideration the subpopulation size that has an effective role in reaching the optimal solution, consider from Figure (٤-٤) that if the size of the subpopulation is more than ٣, this will lead to premature conversion to a solution that may not be the optimal solution; this condition leads the system to local minima.

In the proposed system it has chosen the size of subpopulation equal to '٢', in order to avoid the premature conversion.



Premature conversion

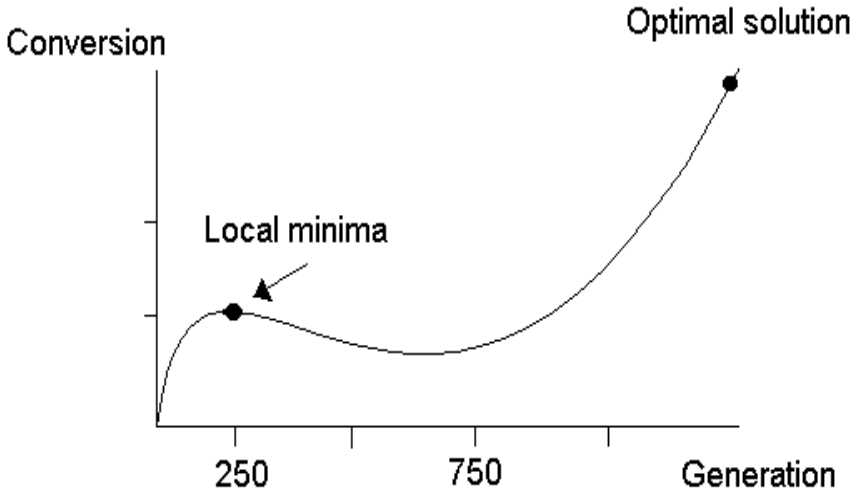


Figure (4-4) Conversion to the Optimal Solution

In general, the Figure (٤-٥) illustrates the selection operation of individuals that share in each of the crossover, mutation, and the evaluation operation in order to obtain the offspring.

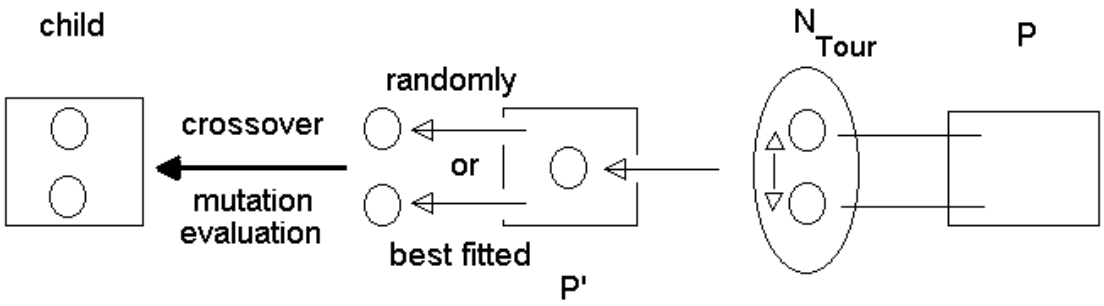


Figure (٤-٥) Tournament Selection

Where is,

N_{tour} : subpopulation

P' : partial population

P : old population

The Figure (٤-٦) illustrates an example of a binary tournament selection. The population consists of a set of chromosomes whose genes are from the English alphabetic *A*.

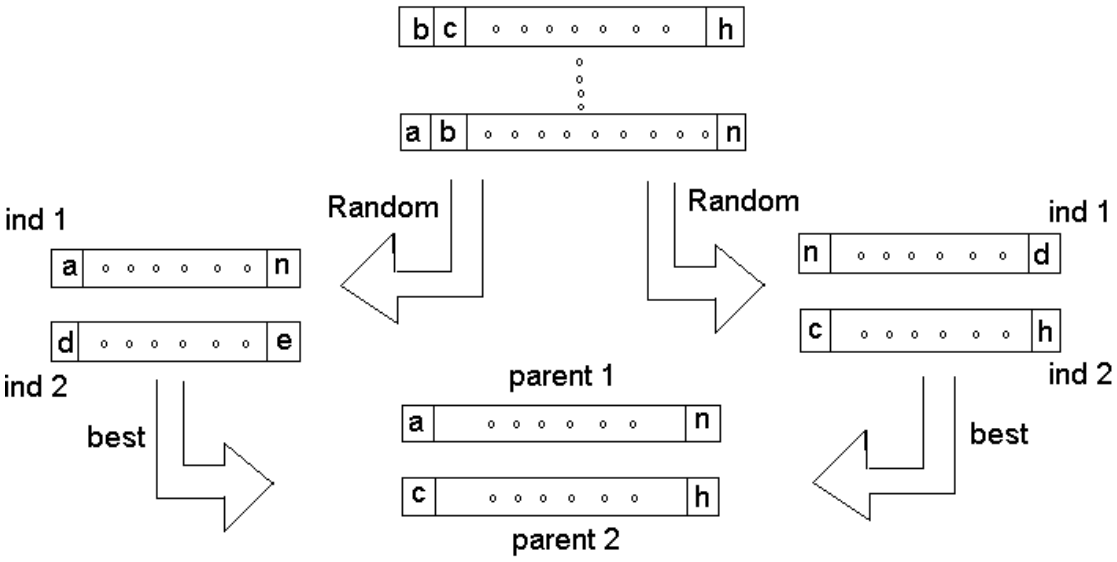


Figure (٤- ٦) Binary Tournament Selection Example

٤.٦ Crossover Operation

In the proposed system, the cycle crossover (CX) has been chosen as one of the permutation crossover operators that mate the matching with the problem. This type of crossover gives a variety of individuals; in addition it avoids the conflict in the genes constructing the chromosome (individual), which is the most important property that must be available in the proposed system.

The general form for the cycle crossover is illustrated in Figure (٤-٧):

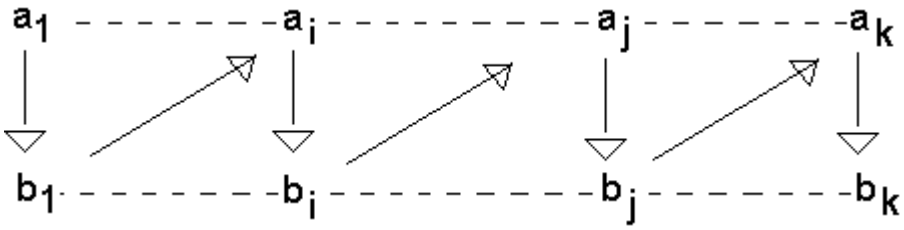
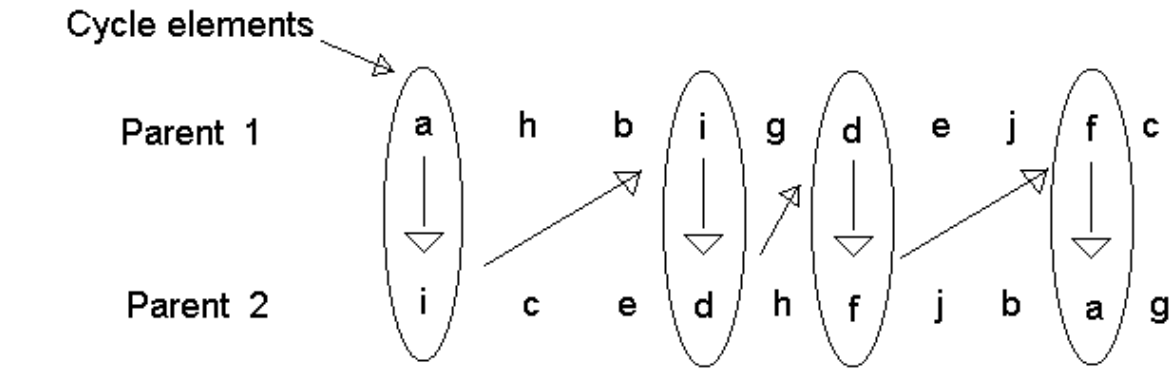


Figure (٤-٧) Cycle Crossover

where a١,bi,aj,ak,b١,bi,bj,bk represents the cycle elements(crossover cycle). The cycle is stopped when $\mathbf{a_{١}=b_{k}}$. The elements that have shared in the cycle keep the same location and the other elements exchange their locations.

The following example in Figure (٤-٨) describes that:



Offspring 1 a c e i h d j b f g

Offspring 2 i h b d g f e j a c

Figure (٤-٨) Crossover Operation

The followed algorithm of the cycle crossover is:

١- Reading the genes (characters) for the parents and their probabilities.

٢- Selecting the first gene of both of the parents, and saving it in the variables 'one','two' respectively.

٣- Comparing 'one' and 'two' variables, if 'one' and 'two' are equal print 'no crossover' and go to step ٦. Otherwise, if 'one' and 'two' variables are not equal , get the similar gene from the first parent for the first gene of the second parent.

٤- Selecting the similar copy gene of the second parent for the selectd gene in step ٣ (first parent), and saving it in 'two' variable.

٥- Comparing 'one' and 'two' variables, if not equal go to step ٣. Otherwise, if equal, exchange the genes that did not share in the cycle, according to the gene and the one corresponding to it.

٦- End.

Mentioning that the crossover rate (Pc) is ١.

٤.٧ Mutation:

From computing the probabilities, it is determined whether there is mutation or not. The mutation probability (Pm) is ٠.٠٠٠٩,if the gene's probability is less than or equal to the Pm then mutation occurs at that gene. On condition mutation occurs, the gene's location that happened at it the mutaion is exchanged with the succeeding gene. If the mutation occurr at the last gene, in this condition this gene's location is exchanged with the first gene.

Figure (٤-٩) illustrates the mutation operation:

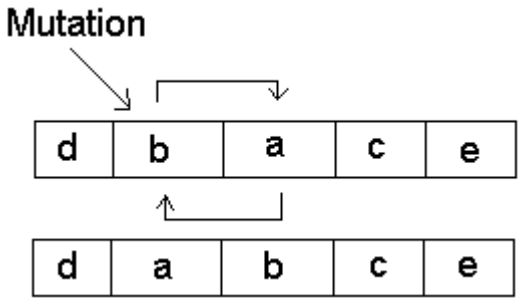


Figure (٤-٩) Mutation Operation

٤.٨ *Evaluate Offspring and Replacement*

The new offspring produced by the crossover operation after building the Huffman tree for each new individual are evaluated and the fitness function is computed.

Then the new offspring is compared with the worst individuals in the population (biggest variance), the offspring are exchanged with the worst individuals in case the offspring is better than or equal to the selected individuals in order to get the best variety in the population.

In other cases, no exchange happens. After finishing the evaluation operation, there will be a new selection and crossover operation again and continue in this way.

Noting that the proposed system had applied the Niching concept, where after beginning the search for the optimal solution, we notice that the system will find peaks of solutions moving between them to reach the optimal solution.

The benefit getting from applying the concept of Niching is providing a set of solution rather than only one solution, as shown in Figure (٤-١٠).

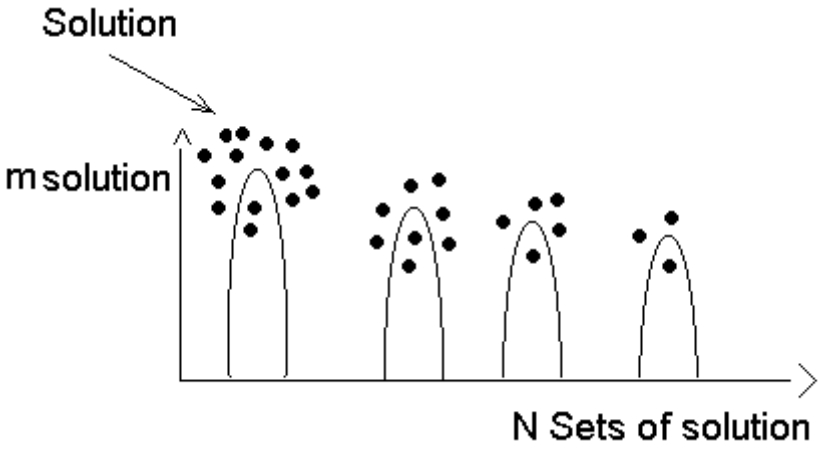


Figure (٤-١٠) Niching Concept (Subpopulation)

٤.٩ *Stop Criteria*

The operation including the crossover operation and generating a new population is continued for some generations (may be thousands of times).

After this, we will consider that one of the individuals with the best fitness (the least variance) prevalent in the population. The rate of prevailing may reach ٥٠٪ or more and this rate is one of the stop criteria; the other stop criteria are the number of cycles (generations) that can reach ١٠٠٠ generation.

As a result, the best individual is the most prevalent in the last population.

٤.١٠ *Compressed Operation*

After finding the best Huffman tree (Dictionary), the text is coded according to the dictionary that has been obtained and according to the codeword for each character in the text. The result is a sequence of '٠' and '١' saved in a file.

The number of zeros are tested, if the number increases ٧٠٪ then the file is inputted to another additional compression algorithm working on binary numbers; this method is Oring bits. This method rises from the compression rate.

After the Oring bits method is applied on the resulting file, the binary numbers are divided to blocks, each block includes eight bits (byte) and is converted to an integer number and saved in byte. The file is sent as a compressed file to other side, mentioning that the dictionary will be associated with the text.

Example :

If we have the following text (file size is ١٠٠ byte):

aaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbccc

ccccccccccccccccccccccccccccccccccccc

After building the following Huffman tree, as shown in Figure (٤-١١):

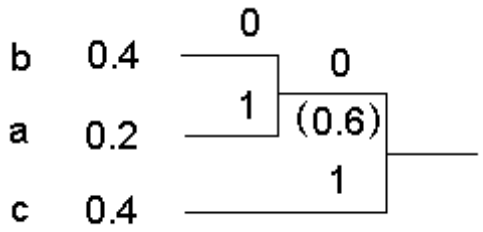


Figure (٤-١١) Huffman Tree

The codeword for each gene is :

c =١ , a= ٠١ , b=٠٠.

The resulted file is :-

٠١٠١٠١٠١|٠١٠١٠١٠١|٠١٠١٠١٠١|٠١٠١٠١٠١|٠١٠١٠١٠١|٠٠٠٠٠٠٠٠|٠٠٠٠٠٠٠٠

|٠٠٠٠٠٠٠٠|٠٠٠٠٠٠٠٠|٠٠٠٠٠٠٠٠|٠٠٠٠٠٠٠٠|٠٠٠٠٠٠٠٠|٠٠٠٠٠٠٠٠|٠٠٠٠٠٠٠٠

٠|٠٠٠٠٠٠٠٠|١١١١١١١١|١١١١١١١١|١١١١١١١١|١١١١١١١١|١١١١١١١١

Noticing from the text that the number of zeros has a high rate, and by that the additional compress using Oring bits is useful, where the compression rate increases from ٧١٪ to ٧٧٪ and as following:-

L١=٠١٠١٠١٠١|٠١٠١٠١٠١|٠١٠١٠١٠١|٠١٠١٠١٠١|٠١٠١٠١٠١|١١١١١١١١|١١١١١
١١١|١١١١١١١١|١١١١١١١١|١١١١١١١١

Oring L١= ١١١١١٠٠٠|٠٠٠٠٠٠٠١|١١١١

L٢= ١١١١١٠٠٠|٠٠٠٠٠٠٠١|١١١١

Oring L٢= ١١١

Then Oring L٢,L٢,L١ is sent with the dictionary through the transmission channel and as follows (file of byte) :-

٣ ٨٥ ٨٥ ٤ ٥ ٩٩ ٩٨ ٩٧ ١٦٠ ٠ ٣

١ ٣١ ١٥ ٨٥ ٢٥٥ ٢٥٥ ٢٥٥ ٢٥٥ ٢٥٥ ٢٤٨ ٨٥ ٨٥

The first byte represents the number of genes that constructs the individual (chromosome length). The second byte and third byte represent the number of bits that represent text length where:

Text length = second byte * ٢٥٥ + third byte, then the text length is equal to ١٦٠.

The next n bytes (n=number of genes) represents the genes (characters). The other next n bytes represent the codeword for each gene (dictionary). while the rest of bytes represent the coding of the text.

٤.١١ *Decompressed Operation*

The first thing done after receiving the compressed file is determining whether the Oring bits have been used or not.

The genes and the dictionary are distinguished from the compressed text, then the text is read bit by bit in a sequential order and is compared with the dictionary in order to get the original text.

In case the file is not inputted to the Oring bits method, then the integer values are converted to the corresponding binary values. So the file is now converted to a sequence of binary numbers (٠ and ١) which represent the text.

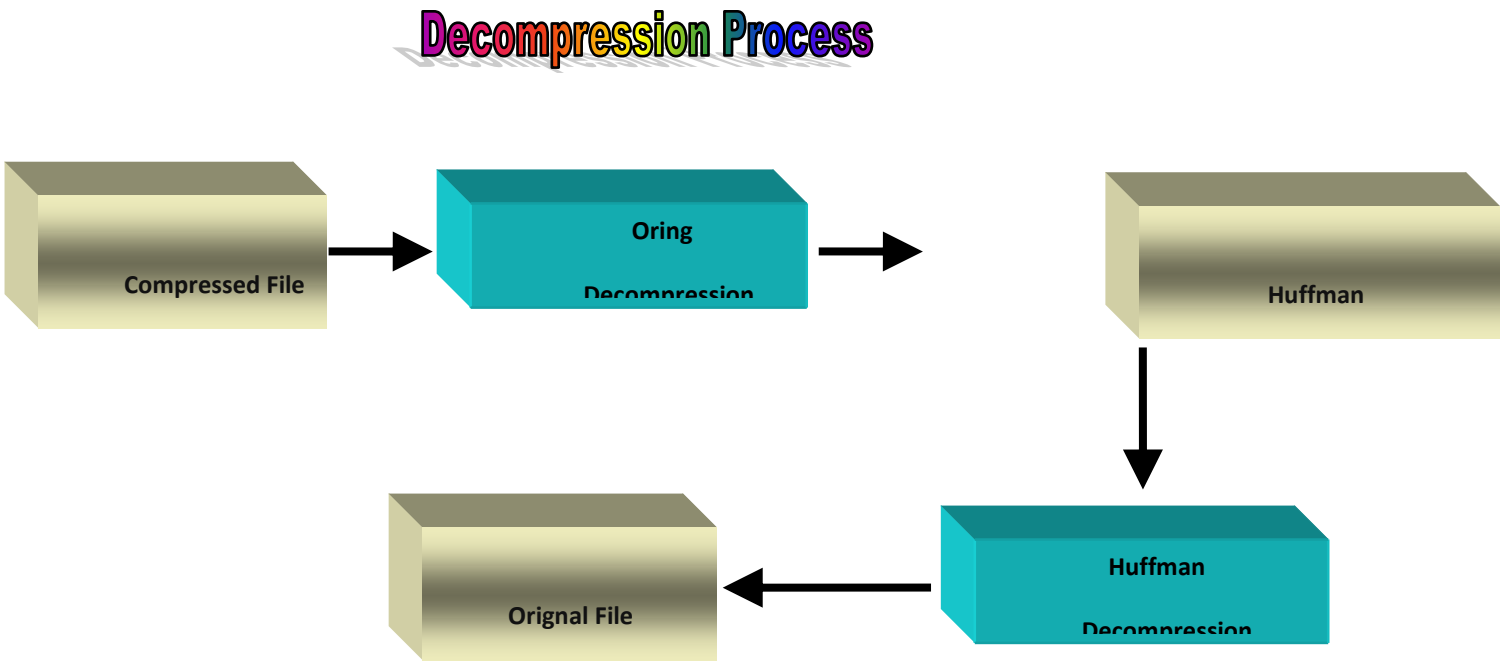
On the other hand, if the Oring bits are used, then the rest of the bytes (text bytes) in the compressed file represent the Oring bits bytes. In this condition the decompression operation begins from the last byte which represents the Oring L٢.

The Oring L٢ is converted to binary values. From the number of one's in the binary value we determine the number of bytes for L٢ which are also converted to binary in order to know the number of one's which give the number of bytes of L١ also converted to a binary value.

After determining each of Oring L٢, L٢, L١, now the Oring bits procedure is applied. As a result, we have a file as a sequence of ٠ and ١, which is the same result of the first case without Oring bits.

The resulting file is the text. The text is read bit by bit in a sequential order and is compared with the dictionary in order to get the original text.

The block daigram below illsturated this prosess :



Figure(٤-١١) Decompression prosess

٥.١ Result Analysis

After building the Huffman tree for the individuals produced from

the genetic algorithm, we have to use a measurement for finding the best tree. In the beginning the Entropy was used to find the best tree. It was found out that all the trees have the same entropy because it depends on the probabilities of the genes that are the same in all the individuals and as a result cannot be used as a measurement for finding the best tree.

Then the variance was used for finding the best Huffman tree. It was found that there was a variety in the variance values that depend on the text itself; and the variety may be large or small. The benefit of the variance depends on the average size of the tree; we will get on codewords with approximate lengths and this will help in the transmission operation where the bandwidth is restricted in the transmission channel and finally making use of the buffer size and transmission time.

It has been noticed that several trees may have the same number of bits constructing them but it has different variance values and the best variance(best tree) is selected which have the codewords with approximate lengths.

These several examples illustrates the individuals and their codewords and also shows the role of the variance in choosing the best Huffman tree.

At first, the number of generation is determined as one of the stop criteria. Then the desired text is inputted as a file and the number of individuals of the initial population are also determined.

Example١:

The number of generation is ٢٠٠ and the number of individuals is ١٥.

abccddeeeeabccddeeeeabccddeeeeabccddeeee…….

The file size is ١٥٠ byte. The initial population is generated as:-

File size :١٥٠

Generation : ١

Ind : ١ Allcode =١٤

Variance =١.٣٦٠

[a] ٠.١٠٠ ٠٠٠٠

[b] ٠.١٠٠ ٠٠٠١

[c] ٠.٢٠٠ ٠٠١

[d] ٠.٢٠٠ ٠١

[e] ٠.٤٠٠ ١

abcde

---

ind : ٢ Allcode =١٤

variance = ١.٣٦٠

[a] ٠.١٠٠ ٠٠٠٠

[d] ٠.٢٠٠ ٠٠١

[e] ٠.٤٠٠ ١

[c] ٠.٢٠٠ ٠١

[b] ٠.١٠٠ ٠٠٠١

adecb

---

ind :٣ Allcode = ١٤

variance = ١.٣٦٠

[c] ٠.٢٠٠ ٠٠٠

[a] ٠.١٠٠ ٠٠١٠

[d] ٠.٢٠٠ ٠١

[b] ٠.١٠٠ ٠٠١١

[e] ٠.٤٠٠ ١

cadbe

The generation operation of individuals for the initial population continues and new individuals are generated with different number of bits for the generated tree that can be better or worse from the previous trees and as follows :

Ind: ٧

allcode: ١٣

Variance: ٠.٩٦٠

[d] ٠.٢٠٠ ٠٠٠

[e] ٠.٤٠٠ ١

[c] ٠.٢٠٠ ٠٠١

[b] ٠.١٠٠ ٠١٠

[a] ٠.١٠٠ ٠١١

decba

After making the crossover operation between the individuals for some generations, a new individual may get produced carrying better characteristics

from the parents and with this it will be the individual with the least variance. The following example illustrates that:

allcode =١٢

Variance =٠.١٦٠

[e] ٠.٤٠٠ ٠٠

[b] ٠.١٠٠ ١٠٠

[d] ٠.٢٠٠ ١١

[c] ٠.٢٠٠ ٠١

[a] ٠.١٠٠ ١٠١

ebdca

Another individual may be produced carrying the characteristics of the previous individual and this individual may be prevalent in the population after some generations and by that it achieves the best individual in the population. This example illustrates that :-

Best tree : ٨

Variance : ٠.١٦٠

[e] ٠.٤٠٠ ٠٠

[a] ٠.١٠٠ ١٠٠

[c] ٠.٢٠٠ ١١

[d] ٠.٢٠٠ ٠١

[b] ٠.١٠٠ ١٠١

eacdb

Table (٥-١) shows the possible variance and Allcode (number of bits for the dictionary) for all the population's individuals for the previous text:-

Table (٥-١) Allcode of Example ١.

Individual	Allcode	Variance
abcde	١٤	١.٣٦٠
decba	١٣	٠.٩٦٠
ebdca	١٢	٠.١٦٠

The relation between the variance and Allcode (number of bits for the dictionary) is shown in Figure(٥-١):

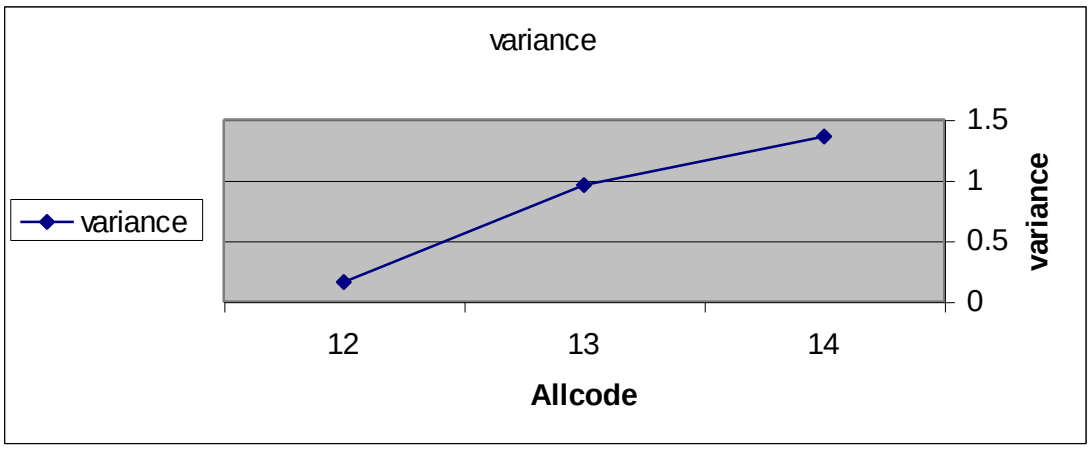


Figure (٥-١) Variance of Example ١.

where the allcode represents the number of bits that the Huffman tree consists of. Noting that the number of the individuals of the initial population has a large and obvious effect on the results where the increase in the number of individuals of the initial population carryout increase in the crossover operation and which finally leads to a variety in the individuals in order to get new individuals and by that avoiding not obtaining a crossover operation that may happen from the fast prevalence for a particular individual.

In addition to the increase in the number of individuals of the initial population, the increase in the variety of the genes of the chromosome (increasing individual's length) leads to the variety in the individuals that will lead to the increase in the crossover operation and finally can obtain the best individual.

Example ٢:

abccddeeeeqwrrttyyyynmlloo…….

Where the file size is ٣٣٨ byte.

The variance table (٥-٢) that is obtained after ٣٠٠ generation is :-

Table (٥-٢)Allcode of Example ٢.

Individual	Allcode	Variance
teydorclmnqwba	٥٤	٠.٢١٣
tyeqorclabnwmd	٥٥	٠.٣٦٧
abcdeqwrtynmlo	٥٦	٠.٥٢١
endmylabwcrotq	٥٧	٠.٦٧٥
mcrloaytdwqnbe	٥٧	٠.٨٢٨
elnrtwbmaoqcdy	٥٨	٠.٩٨٢

As it is noticed from the Table (٥-٢), the same allcode value gives two different variance values. In this case, the structure of the tree has an effect on the codeword length and finally has an effect on the variance.

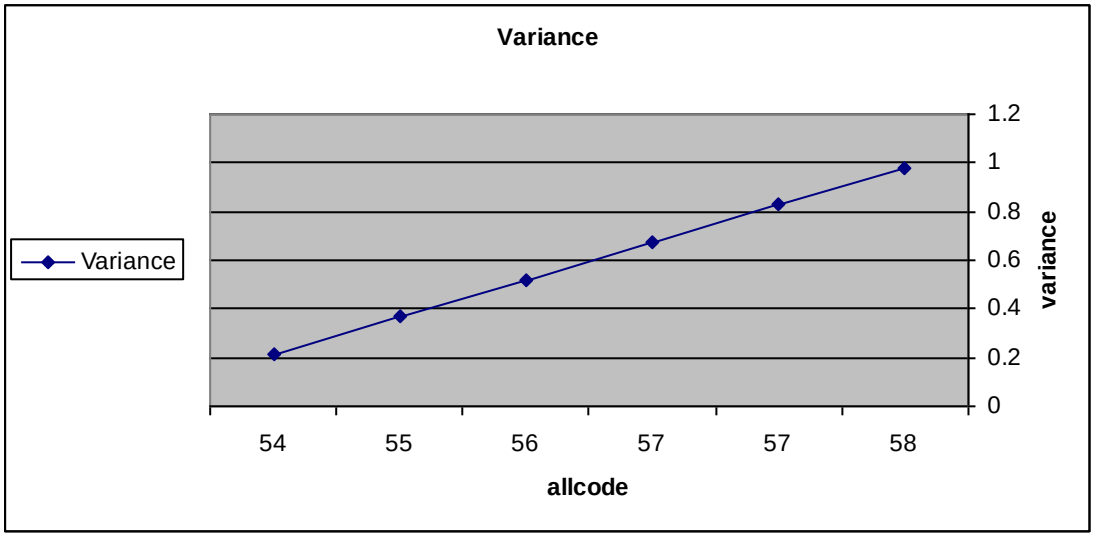


Figure (٥-٢) Variance of Example ٢.

The randomize plays a large role in obtaining the results and the speed in reaching the optimal result.

Example ٣:

The boy is very clever and happy

Where the variety in the variance that can be obtained is shown in table (٥-٣):

Table (٥-٣) Allcode of Example ٣.

Individual	Allcode	Variance
ahvdecynrtpoilsb	٦٦	٠.٤١٠

cedsrbpinvhyalto	٦٧	٠.٥٦٤
theboyisvrclandp	٦٨	٠.٧١٧
yvplronchtasdieb	٦٩	١.٠٢٥

Noting that after ٣٠٠ generations we can obtain the best variance that is ٠.٠٤١ .

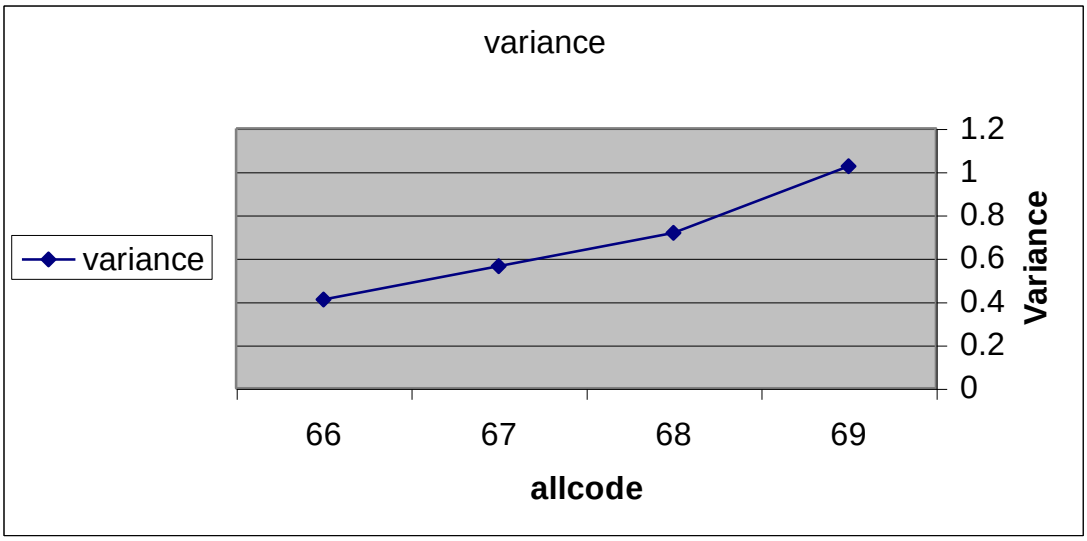


Figure (٥-٣) Variance of Example ٣.

The proposed system applied on a set of files has been chosen because the variety in the variance value in order to clear all the possible situations. It has been showed the effect of each of the file size, chromosome length, and probability of genes on the compression ratio as showed in the following tables. The Oring bits have been applied on a set of these files showing its effect in increasing the compression ratio.

Table (٥-٤) illustrates a set of examples taking into consideration different file sizes, while the chromosome length, population size, and generation number are fixed.

Table (٥- ٤) Effect of Different File Size

	File name	File size	Chrom. Length	Population size	Generation No.	Compression Ratio
١	File ١	١٥٠B	٥	١٥	٢٠٠	٦٣.٣٣٣٪
٢	File ٢	٣٠٠B	٥	١٥	٢٠٠	٦٨.٠٠٠٪
٣	File ٣	٦٠٠B	٥	١٥	٢٠٠	٧٠.٣٣٣٪

Table (٥-٥) illustrates a set of examples taking into consideration different chromosome lengths, while the other parameters are fixed.

Table (٥- ٥) Effect of Different Chromosome Length

	File name	File size	Chrom. Length	Population size	Generation No.	Compression Ratio
١	File ٣	٦٠٠B	٥	٢٥	٣٠٠	٧٠.٣٣٣٪
٢	File ٦	٦٠٠B	١٠	٢٥	٣٠٠	٥٦.٥٠٠٪
٣	File ١١	٦٠٠B	٢٠	٢٥	٣٠٠	٤٠.٣٣٣٪

Table (٥-٦) illustrates the effect of the probability of genes on the compression ratio. The increment in the probability of genes gives the gene the shortest codeword and as a result increases the compression ratio.

Table (٥-٦) Effect of Probability of Gene

	File name	File size	Chrom. Length	Population size	Generation No.	Compression Ratio
١	File ١	١٥٠B	٥	١٥	٢٠٠	٦٣.٣٣٣٪
٢	File١٢	١٠٠B	٥	١٥	٢٠٠	٧١.٠٠٠٪
٣	File١٠	٩٠B	٥	١٥	٢٠٠	٧١.١١١٪

After applying the Oring bit algorithm on some of the above files, taking into consideration that the ratio of zeros in the file must be above ٦٥٪ in order to get on good results.

The files that satisfied the condition and gave good results are illustrated in Table (٥-٧).

Table (٥-٧) Results of Oring Files

	File name	File size	Chrom. Length	Compression ratio without Oring	Compression ratio with Oring
١	File ٦	٦٠٠B	١٠	٥٦.٥٠٠ %	٦٢.١٦٧ %
٢	File ١٠	٩٠B	٥	٧٠.١٠٣ %	٧٧.٣٢٠ %
٣	File ٩	٨٨٤B	٣	٧٩.١٨٦ %	٨٤.٦١٥ %

Also the proposed system has been applied on segments from DNA file taking different file sizes and results are viewed as follows:

Table (٥-٨) Results of DNA Files

File name	File size	Compression ratio
DNA ١	٧٢ B	٥٩.٧٢٢٪
DNA ٢	١٥٣ B	٦٧.٣٢٠٪
DNA ٣	٣٠٠ B	٧١.٣٣٣٪
DNA ٤	٥٤٢ B	٧٢.٨٧٨٪

It has been noticed that there was not a large variety in the individuals because of the small number of genes (only four genes). It has also been noticed that it was difficult to get good results from applying the Oring on the DNA file because there was no occurrence of a large sequence of single genes.

٥.٢ Conclusions

١. The subpopulation size has an effective role in reaching to the optimal solution; if the size of the subpopulation is more than ٣, this will lead to premature convertion to a solution that may not be the optimal solution.

٢. The same allcode value gives two different variance values. In this case, the structure of the tree has an effect on the codeword length and finally has an effect on the variance.

٣. The number of individuals (population size) has an effect on obtaining the best tree where the small population size leads to a small variety and as a result leads to small crossover operations between the individuals because of getting the same first gene. As a result, the best individual (tree) will not be obtained.

٤. The number of generations has an effect on reaching to the best Huffman tree where continuing in selecting various individuals gives a larger chance in selecting the best possible individual.

٥. In order to get on the best compression for the Oring method when merging the Huffman compression method and Oring, the character with the largest probability takes the value '٠' instead of the value '١' in building the Huffman tree.

٦. The chromosome length has an effect on the compression ratio. Whenever the chromosome length was longer, the compression ratio is smaller. This happens because the increase in the size of the Huffman tree causes as a result an increase in the number of bytes transmitted through the transmission channel.

٥.٣ Future Works

١- Applying the meta -Genetic Algorithm on Genetic Algorithm (Optimization of GA).

A meta-genetic algorithm has been used to optimize the genetic algorithm for cell placement. The three parameters optimized are the crossover rate, inversion rate and mutation rate. The meta-genetic algorithm is itself a genetic optimization process which runs the genetic algorithm to solve a placement problem and manipulates the genetic parameters to optimize the fitness of the genetic algorithm. The individuals in the population of meta-genetic algorithm consist of three integers in the range [٠,٢٠], representing the mutation rate, inversion rate, and crossover rate for the genetic algorithm.

٢- Using Breeder Genetic Algorithm (BGA):-

BGA represents a class of random optimization techniques gleaned from the science of population genetics, which have proved their ability to solve hard optimization problems with continuous parameters. BGA which can be seen as a recombination between Evaluation strategies (ES)and Genetic Algorithm(GA), uses truncation selection which is very similar to the (u ,λ)strategy in ESs and the search process is mainly driven by recombination making BGAs very similar to GAs. It has been proven that BGAs can solve problems more efficiently than GAs due to the theoretical faster convergence to the optimum and they can, like GAs, be easily written in a parallel form.

٣- Applying the compression algorithm, the (Delta algorithm) on the research results for decreasing the cost of communication channels (reducing band width).

After finishing the compression operation and converting the sequence of binary numbers to integer numbers and before saving it as bytes, the delta algorithm is applying in order to reduce the value of the integer numbers and finally reducing the amplitude for the transmitted signal and this gives us a small band-width.

٤- The proposed system can be applied on images by dealing with an image as segments where Huffman algorithm is applied on each segment.

REFERENCES

A. Joglekar & others, "**Genetic Algorithms and Their** [١]

Use in the Design of Evolvable Hardware", Fr. Conceicao Rodrigues College of Engineering, Bandra, Mumbai - ٤٠٠ ٠٥٠,IEEE, ١٩٩٧.

A. Khelfa , "**A Genetic Clustering for Image Segmentation "** , [٢]

M.SC Thesis, April ٢٠٠٢.

A. Armstrong and others, **" Genetic Algorithms for Image** [٣]

Compression ", ٢٠٠١ .

A. W. berger and others," **A Hybrid Coding Strategy for Optimized Test** [٤] **Data Compression",** University of Innsbruck, Austria, Proceedings IEEE International Test Conference, Charlotte, NC, USA, September ٣٠ – October ٢, ٢٠٠٣.

[٥] B. J. Schachter and others," **Some Experiments in Image**

Segmentation by Clustering of Local Feature Values ", Pattern Recognition, ١١, ١, ١٩, ١٩٧٨.

[٦] B. D. Davison and others, " **Effect of Global Parallelsim on A**

Steady State GA ", Proceedings of The Evolutionary Computing and Parallel Processing Workshop (GECCO'٩٩), Orlando, ١٩٩٩.

D. Whitley, "**A Genetic Algorithm Tutorial** " , Computer [٧]

Science Department Colorado State University, Fort Collins CO.,

١٩٩٤.

[٨] D. Salomon , " **Data Compression the complete reference** "
spring verlag Newyourk ,USA , ١٩٩٨ .

[٩] F. Hansson ",**Lempel-Ziv-Welch and Huffman compression ",**
٢٥ January ٢٠٠٤ .

[١٠] G.B. Parker and others, "**Evolving Hexapod Gaits Using a Cyclic Genetic Algorithm " ,** Department of Computer Science Indiana University Bloomington, ١٩٩٨.

[١١] Goldberg D. E. , **" Genetic Algorithms in Search, Optimization and Machine Learning "** ,Addison-Wesley , ١٩٨٩ .

[١٢] G. Kempe, "**Computer Science Honours Research Report** *Compression and Computational Gene Finding"*, ١ ***November*** ٢٠٠٢.

[١٣] G. E. Blelloch "**Introduction to Data Compression**" Computer Science Department Carnegie Mellon University , October ١٦, ٢٠٠١.

[١٤] H. J. Martinez and others, " **Evolution of Cellular Automata for Digital Image Compression",** de Ingenieria Universidad Central de Venezuela Caracas, Venezuela, ٢٠٠٦.

[١٥] From Wikipedia, the free encyclopedia "**Huffman coding** ", GNU Free Documentation license, January ١٩,١٩٩٦.

I. Harvey ," **The Microbial Genetic Algorithm ",**School of [١٦]
Cognitive and Computing Sciences ,University of Sussex ,
Brighton BN١ ٩QH, UK, inmanh@cogs.susx.ac.uk.,١٩٩٦.

J. Socha, "**Efficient Meta-Heuristics for Lossless Data** [١٧]
Compression", School of Computer Science University of
Waterloo, Waterloo, Ontario, ٢٠٠٣.

J. M. Pullen," **Data Compression, Security Principles** [١٨]
Integrity, Appropriate Use ",٢/٣/٠٣ © ٢٠٠٣.

[١٩] J. Smith & others, " **Replacement Strategies in Steady State**
Genetic Algorithms : Static Environment ", Foundation of
Genetic Algorithms V, ٢١٩, ١٩٩٨.

[٢٠] M. Obitko **,"Introduction to Genetic Algorithms with Java**
Applets",١٩٩٨.

M. Sasikumar , "**Genetic Algorithms** ",NCST, Bombay. [٢١]

P. M. Elizabeth and others, "**Genetic Algorithm for VLSI** [٢٢]
Design ,layout & test Automation ", ١٩٩٩, prentice Hall ,
PTR USA.

[٢٣] P. Busetti ,"**Genetic algorithms overview**",١٩٩٥.

R. C. Gonzalez and others, "**Digital Image Processing"**, [٢٤]
University of Tennessee, ١٩٩٢.

R. He ,"**Indexing Compressed Text",** A thesis,Waterloo, Ontario, Canada, ٢٠٠٣ [٢٥]

R. Müller , **"Image Compression"** Part II: Image Processing Computer Graphics and Image Processing , Winter Semester ٢٠٠٣/٠٤. [٢٦]

R. A. Watson and others, **"Recombination Without Respect: Schema Combination and Disruption in Genetic** ***Algorithm Crossover " ,*** *Volen Center for Complex Systems, Brandeis University, Waltham,* ٢٠٠٠. [٢٧]

Sheila Horan,**"Data Compression Statistics and Implications ",** New Mexico State University, chapter ٢٧- Data compression, ١٩٩٧. [٢٨]

S. W. Smith, A sample chapter from: **The Scientist and Engineer's Guide to Digital Signal Processing",** ١٩٩٧ . [٢٩]

[٣٠] T. A. Abbas Al- Asadi "**A Hybrid Algorithm For Images Compression**" , Ph.D. Thesis, Computer Science , University of Technology, ٢٠٠٤ .

[٣١] T. H. Al-Kafaji **"Improving Document Retrieval Using Genetic Algorithm"** , M.SC. Thesis, September, ٢٠٠٠.

U. Aickelin **, "Solving Multiple-Choice Problems with Genetic Algorithms**", School of Computer Science University of Nottingham UK uxa@cs.nott.ac.uk. [٣٢]

W. M. Spears **"Adapting Crossover in a Genetic Algorithm " ,** Washington, D.C. USA , ١٩٩٥. [٣٣]

W. KEONG NG," **Lossless and Lossy Data Compression "** ,Nanyang Technological University ,Singapore, ١٩٩٦. [٣٤]

W. J. Mason and others ,**" Optimal Earth Orbiting Satellite Constellations Via a Pareto Genetic Algorithm"** Department of Aeronautical and Astronautical Engineering University of Illinois at Urbana-Champaign Urbana,١٩٩٨. [٣٥]

From Wikipedia, the free encyclopedia," **Genetic algorithm" ,** GNU Free Documentation license, June ٣٠, ٢٠٠٥. [٣٦]

REFERENCES