Image Compression using Genetic Algorithm

Ali K. Idrees , Suhad A. ALI, Esrra H. Obead
Department of Computer Science, College of Science for Women, University of Babylon

Abstract:

In this paper, A steady – state Genetic Algorithm (SSGA) based two phase image quantization algorithm for image compression is proposed. The aim of the first phase is create the initial codebook that used to categorize the image blocks by a according to some distortion measures, while the SSGA will produce the best representative Block that represents the all blocks in each unit within the codebook in the second phase. The simulation results explain that the proposed SSGA exhibits a good compression ratio with high quality reconstructed image.
Keywords: image compression, vector quantization, Genetic Algorithms

الخلاصة

أُقترح في هذا البحث، طريقة تستخدم الخوارزمية الجينية (SSGA) لضغط الصور باستخدام المكمم ألاتجـاهي بمـرحلتين. إن الهدف من المرحلة الأولى هي لخلق codebook ابتدائي يستخدم لتصنيف بلوكات الصورة وفقا لمقياس خطا معـين، بينمـا تـستخدم خوارزمية SSGA للحصول على أفضل تمثيل للبلوك الذي يمثل كل البلوكات الموجودة في كل وحدة في داخل الـ codebook فـي المرحلة الثانية. توضح النتائج التي تم الحصول عليها باستخدام الطريقة المقترحة بان نسبة الضغط جيدة مـع جـودة عاليـة للـصورة المسترجعة.

1- Introduction:

Vector quantization is an effective technology for data compression and it has been successfully used in speech encoding and image compression (Salmon, 2007) . The basic idea behind all vector quantization based image compression techniques is similar and we can define as follow: Given an image, we divide it into small blocks of pixels typically 2×2 or 4×4. Each block is considered a vector. The encoder maintains a list (called codebook) of vector and compresses each block by writing on the compressed stream a pointer to the block in the codebook. The decoder has the easy task of reading pointers, following each pointer to a block in the codebook, and joining the block to the image – so – far, were the decoder will receive the pointer numbers across the channel (Salmon, 2007; Cosman et. al. , 1996). The design of vector quantizers that yields the lowest distortion is one of the most challenging problems in the field of source coding. However, this problem is Know to be difficult (Gersho et. al. ,1992).

many traditional search method can hardly find the best representative of data especially when the data become large. To determine the closest ones are costly in terms of processing time and memory. The conventional solution technique work through a process of iterative refinements which yield only locally optimal results.

Po (1990) Attempt to reduce the computation time by reducing the dimension measure in the LBG algorithm, but the codebooks generated by this method is slightly degraded. Ma et. al. (1991) propose a maximum descent algorithm that compared with LBG algorithm, the codebook performance is improved and the computation time is reduced. However, this algorithm can only obtain a local optimal code bock. Torres. and Huguet (1994) and Ra and Kim (1993), They are makes some slight modification in the LBG search to reduce the computation time , however they cannot improve the codebook performance.

The previous work in image compression based on Genetic Algorithms is as follow: Ng et. al. (1995) design three versions of Genetic algorithms based on Generalized Lloyd Algorithm (GLA) that called Genetic GLA (GGLA) for computing vector quantizers . In their work, each chromosome represents a codebook and each gene is a codevector. The population is a set of code bocks. This will lead to more complexity and the GGLA costs more memory and computation time as the population size increase. Delport and Koschorreck (1995) and Pan et. al. (1995) proposed two Genetic algorithms for codebook generation problem in vector quantization, but these algorithms, however , are not pure genetic algorithms but hybrid combinations of GA and GLA . Moreover, the chosen parameters of GA are not analyzed in these studies, thus the question of the real benefits of genetic algorithms should still be considered. Fränti et. al. (1997) used a hybrid combinations of GA and GLA for codebook generation problem, where each chromosome represents a codebook and each gene is a codevector. Their proposed system need more time and more memory as the population size increase. Further more, the hybrid system need larger possible iteration for GLA to decrease the distortion value that lead to more complexity for the GA. Lu et. al. (1999) used a GA for a codebook design of vector quantization. Where each chromosome represents a codebook and each gene is a codevector. Their proposed algorithm need more processing time and more memory as the population size increase. Their proposed GA suffers of prematurely phenomenon and may lack in local optimum during the search. Merlo et. al. (1999) analyzes the image compression problem using genetic clustering algorithm. They are concentrates in their paper on finding an algorithm that perform the clustering efficiently. They are suggested four clustering algorithms, but don’t explain how they can use these algorithms in image compression. Armstrong and Jiang (2001) used a simple genetic algorithm for image quantization to find near – optimal codeword. their simple GA produce the codevector that represent the all codevectors belong to the same index in the codebook, but their algorithm suffer of prematurely and may lack in local optimum during the search because selecting the best chromosomes at each generation for crossover with crossover probability very low and very low mutation probability That lead to remain the same two chromosomes the best to be selected again in the next Generation.

Our proposed system used a Steady – State Genetic Algorithm (SSGA) based two phase image quantization for image compression, the aim of the first phase is to create the initial codebook that used to categorize the image blocks by, according to some distortion measures. While the SSGA will generate the best reprehensive block that represent the all blocks in each unit within the codebook, in the second phase. According to the results obtained in our experiments we think that the SSGA solution to the image quantization problem is very promising.

The organization of this paper is as follows. In section 2, we shows the genetic algorithm and its role in solving optimization problem. Section 3, describes the configuration of our SSGA based two phase image quantization. The simulation results are explained in section 4. Finally, the conclusion and future work is explained in last section.

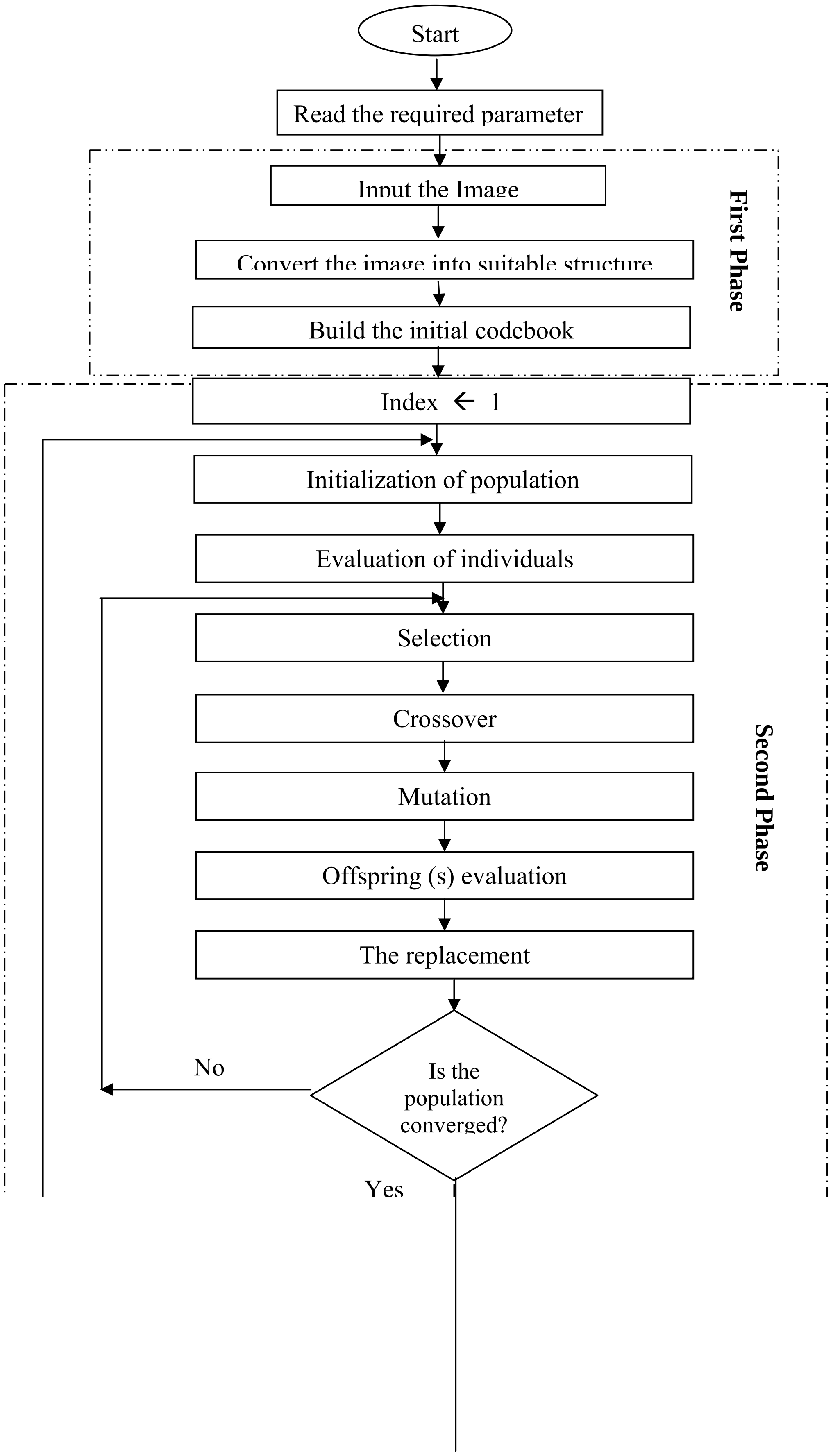
2. The genetic algorithm for optimization:

The GAs are search algorithms that simulate the process of natural selection and survival of fitness during the search process , it can automatically achieve and accumulate the knowledge about the search space, and adaptively control the search process to reach the overall optimal solution (Goldberg, 1989; Hourani, 2004).

GA has been quite successfully applied to optimization problems. It belongs to the class of probabilistic algorithms, yet they are very different from random algorithms as they combine elements of directed and stochastic search. Because of this , GAs are also more robust than existing directed search methods. The Genetic based search methods maintain a population of potential solutions, while all other methods process a single point of the search space (Michalewicz,1996) GAs are gradient-free, parallel optimization algorithms that use a performance criterion for evaluation and a population of possible solutions to the search for a global optimum. GAs are capable of handling complex and irregular solution spaces, and they have been applied to various difficult optimization problems (Chambers, 2001). Generating the best representative block that represent the all blocks in the same unit (cluster of blocks) in the codebook (a set of N units) can be formulated as an optimization problem; we use the SSGA to solve it for image compression.

3. The SSGA based Two-phase image quantization:

The flowchart in figure (2) explain the main steps in the proposed system for image compression


Start
Read the required parameter
Input the Image
Convert the image into suitable structure
Build the initial codebook
First Phase
Index ← 1
Initialization of population
Evaluation of individuals
Selection
Crossover
Mutation
Offspring (s) evaluation
The replacement
Is the population converged?
No
Yes
Second Phase

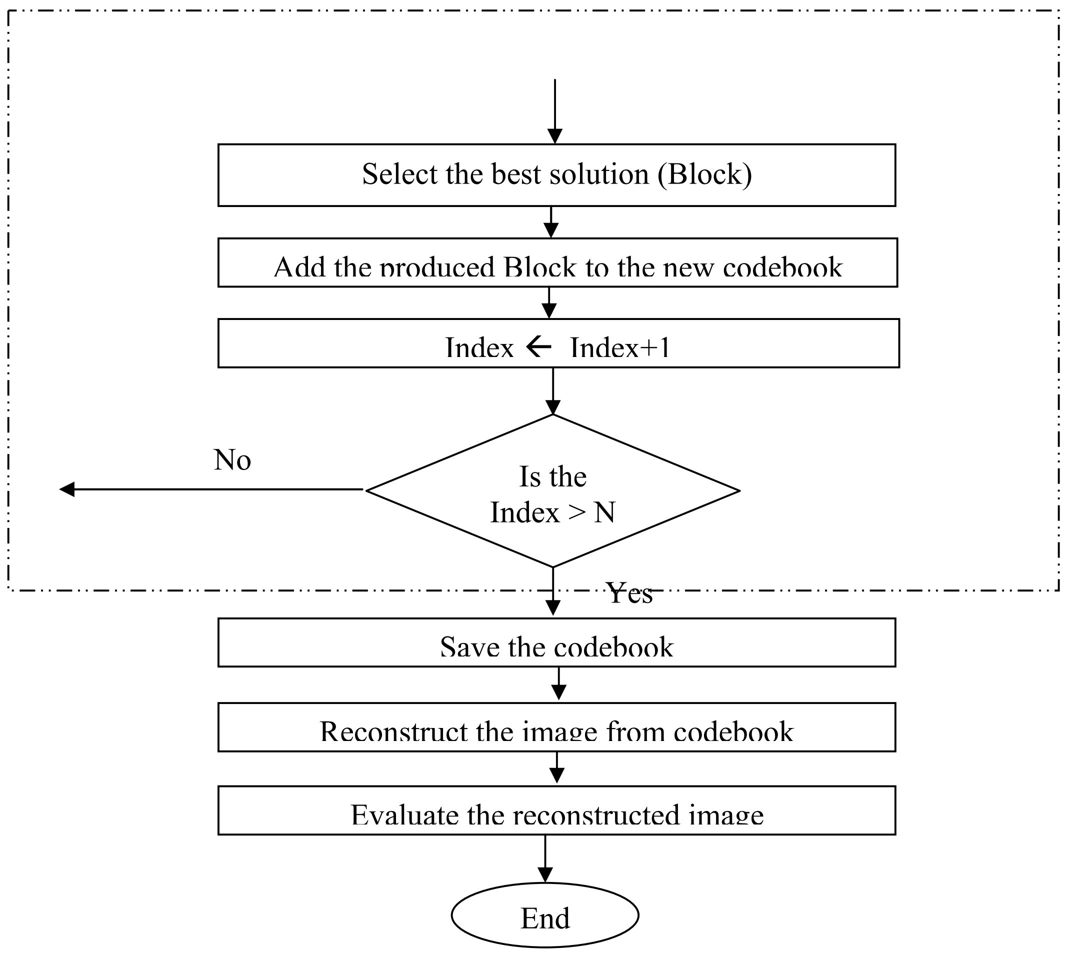

Select the best solution (Block)
Add the produced Block to the new codebook
Index ← Index+1
No
Is the Index > N
Yes
Save the codebook
Reconstruct the image from codebook
Evaluate the reconstructed image
End

Figure (2): Flowchart of proposed system for image compression

We will explain the steps of the flowchart in more detail as follow:

1- read the required parameters: we need at the first , read the important parameters that used by the proposed system such as, population size (popsize), maximum number of Generation (MaxGen), chromosome length (chromlen) that equal to the block size , crossover probability (pc), mutation probability (pm) and the image file that required to be compressed.

2- Input the image: This step include reading the data from BMP image file, where we use a 256 gray level and color images with size (512 * 512) pixel.

3- Convert the image into a suitable structure: we will divide the data of the image into blocks of pixels, where the size of the block is an important and it affects on both the image quality and compression ratio. For larger block size will increase the compression ratio and decrease the image quality (and vise versa). We made balancing between the quality and the compression ratio by taking (4 *4) pixel image blocks that the SSGA operate on it.

4- Build the initial codebook: In this step, we split the overall image blocks into N units as in (Armstrong and Jiang, 2001) with some modification added after classification; each unit is made up of several labels which belong to this unit. N codewords are used to represent these N units where this N codeword are selected randomly from the image blocks. When the image is quantized we require that each 4*4 image block only reference it's codeword using 8-bit. An 8-bits reference index allows 256 representative blocks in the codebook. This initial codebook that contains 256 codeword that selected randomly from the image are used to categorize the image blocks by the following algorithm

```
Algorithm ImageBlock-Classifier
   For I ⟵ 1 to M
      Hpsnr ⟵ 0
      For J ⟵ 1 to N
          Tpsnr ⟵ Function-PSNR (Blocks (I), codebook(J))
          If Tpsnr > Hpsnr Then
             Hpsnr ⟵ Tpsnr
             BestUnit ⟵ J
          End if
      Next J
   Increase the number of blocks that ∈ codebook (BestUnit ) by 1
    Save the Image block position I into array of block position that ∈ codebook
        (BestUnit )
  Next I
End of Image Block-Classifier
```

Where M represent the overall number of blocks into the image , and N is the number of units into the codebook.

The function –PSNR used as a similarity measure between the image block represented as blocks (I) and the initial representative codeword of each unit, codebook(J). Each block in image will take for it the PSNR with each of N initial representative blocks in the codebook, where the image block will belong to the unit that give high PSNR.

We need to calculate the Avgcodebook that contain N codewords, each of it represents the center vector of all vectors that belong to the same unit. Avgcodebook can be calculated according to the following formula

$$\text{AvgCodebook}_r = \frac{\sum_{i=1}^{n^r} x_{a_i^r}}{n^r} \qquad \text{……….} (1)$$

Where r ≡ the number of units in the codebook (r = 1,2,..,N).

n^r ≡ the number of blocks in unit r

x a_i^r ≡ the vector whose label is a_i^r

and $n^1 + n^2 + \ldots + n^N = M$.

5- set the index to 1 , where the index is a counter for the number of unit in the codebook.

6- initialization of population : The SSGA will operate on each unit in the codebook to produce the best representative block that represent the all blocks in the unit. The first step in SSGA is initialize a population that selected randomly from the blocks that belong to the unit that currently SSGA operate on. The number of individuals equal to the (popsize -1) plus the codeword in the Avgcodebook. Each individual in the population consists of a chromosome , diff-pixel , and fitness. Each chromosome represents a block of image (4*4 pixel) and each gene represents the actual value of the pixel.

7- Evaluation of the individuals : each individual is evaluated by using fitness function. The chromosome that have high fitness value will be the best representative block for it's unit. For each individual, we need an array called diff-pixel that contain on the cumulative differences of opposite pixel of all blocks in the same unit to be used in fitness calculation algorithm. The diff-pixel array is calculated according to the following formula:

$$diff - pixel_i^r = \sum_{j=1}^{n^r} \left| P^i - X^i_{a_j^r} \right| \quad \ldots\ldots.. \quad (2)$$

Where diff-pixel : represents an array of cumulative difference of opposite pixels of all blocks in the same unit , r = 1,..N , i= 1,…, chromlen.
P^i : represents the chromosome with gene i.

X : represents the block with value i in unit r. where this block it's label a_j^r

n^r : the number of blocks in the unit r.

The first value in the diff-pixel array is calculated by taking the cumulative difference between the first gene value in P with the first value in each block in the same r. The second value in diff-pixel is calculated by taking the cumulative difference between the second value in P with the second value of each block in unit r and so on. This lead to each individual will have a diff-pixel that used to calculate the fitness value of the individual as in the following algorithm.

```
Algorithm fitness-calculation
  For i ⟵ 1 to popsize
      Counter ⟵ 0
      J ⟵ 1
      Do while ( J $\le$ popsize ) and ( I $\neq$ J )
          For k ⟵ 1 to chromlen
              If diff-pixel $^i_k$ < diff-pixel $^j_k$ Then
                 Counter ⟵ Counter + 1
              End If
          Next k
      Loop
```

```
            Fitness_i ⟵ counter / chromlen
        Next i
    End of Fitness-calculation
```

8- Selection: in this step, we select two parents for crossover and to produce new offspring(s). We study in these paper three selection methods such as:
   - Best individual selection (BIS) : this method of selection select the best chromosome in the population that has a larger fitness as a first parent. If the BIS used to select the second parent, it will select the chromosome that has a second larger fitness.
   - Binary Tournament Selection (BIS): In this method, two different individuals are selected randomly. The individual that has higher fitness will win to be added to the crossover mate. If the fitness value of the first individual is equal to the fitness value of the second individual, one of them will be selected randomly (Michalewicz, 1996; Schmidt. and Stidesen, 1997).
   - Hybrid selection (HS): In this method, the first parent is selected by the BIS, and the second parent is selected by the BIS.

9- Crossover: The crossover property recognized the GA from other evolutionary algorithms (evolutionary strategy and evolutionary programming) (Schmidt and Stidesen, 1997). Crossover is the process of exchanging the parents' genes to produce one or two offspring that carry inherent genes from both parents. We used in this study eight crossover methods that are:
   - Single point crossover (1x): It operates by picking a random point within the two parent chromosomes, and then exchanging the genes of the two chromosomes at or below this point to produce two new offspring(Goldberg, 1989; Michalewicz, 1996). This 1x occurs with a certain crossover probability; otherwise the two parents are copied as offspring.
   - Two point crossover (2x): It operates by picking two random points within two parent chromosomes, and then exchanging the genes between these points in each parent to produce two new offspring. This is performed with a certain probability, otherwise the two parents are copied as offsprings (Goldberg, 1989; Deb, 1998).
   - Uniform crossover (UX): Genes are randomly copied from the two parents to compose two new offspring with a probability P_u (p_u=0.5) (Deb, 1998; Chambers,2001).
   - Modified single point crossover (M1x): It operates by picking a random point within the two parent's chromosomes, and then copy the first part from the first parent and the second part from the second parent to the only one new offspring with a certain crossover probability.
   - Arithmetical Crossover: It is a weighted average of two vector (chromosomes) x1, and x2 that defined as follow:

$$\tilde{x}_1 = \lambda_1 x_1 + \lambda_2 x_2 \quad \text{......................(3)}$$

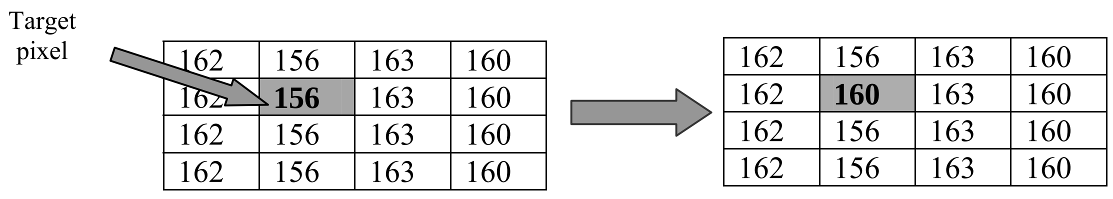
$$\tilde{x}_2 = \lambda_1 x_2 + \lambda_2 x_1 \quad \text{......................(4)}$$

$\tilde{X}_1$ and $\tilde{X}_2$ are offsprings and $\lambda_1 = \lambda_2 = 0.5$ (Hourani, 2004; Gen, and Cheng, 2000).

- Blend Crossover: It creates offspring randomly with a hyper-rectangular define by the parent points. Suppose that the first parent has the value P1, the second parent p2, and that p2 > p1. Let I= $|P_1 - P_2|$ and α > 0, ß < 1 ,then an offspring is generated by randomly choosing a point within the interval [P_1- αI , P_2 + ßI], where α = 0.5, and ß = 0.5 . (Gen, and Cheng, 2000).
- Sphere Crossover : It produces one offspring (Z_1,..,Z_n) from two parents (X_1,…,X_n) and (Y_1,…,Y_n) as follow:
  $Z_i = \sqrt{\alpha X_i^2 + (1-\alpha) Y_i^2}$ ………..(5) , I = 1, 2,……, n
  Where α is a random number chosen in [0..1]. (Gen, and Cheng, 2000).
- Direction-Based Crossover : This method uses the value of the objective function in determining the direction of genetic search. The operator generate a single offspring $X^{\sim}$ from two parents X_1 and X_2 according to the following rule
  $X^{\sim} = r * (X_2 - X_1) + X_2$ ……(6).
  Where r is a random number between 0 and 1. It also assumes that the parent X_2 is not worse than X_1 that is $f(x_2) \geq f(x_1)$ for maximization problems (Gen, and Cheng, 2000).

10- Mutation: It takes place after crossover is performed. Mutation randomly changes a selected gene in the offspring. This occurs at an assigned rate, usually very infrequently. The purpose of mutation is to prevent falling into a locally optimal solution of the solved problem (Hourani, 2004). We use, in this study, two proposed mutation methods:

- Mutation 1: This method deals with the chromosome as a block of n * n pixel. It select randomly one gene (pixel) in this chromosome (block) and then replaced by the average values of the neighbor pixels with it's value(Armstrong and Jiang, 2001), as in the following figure :



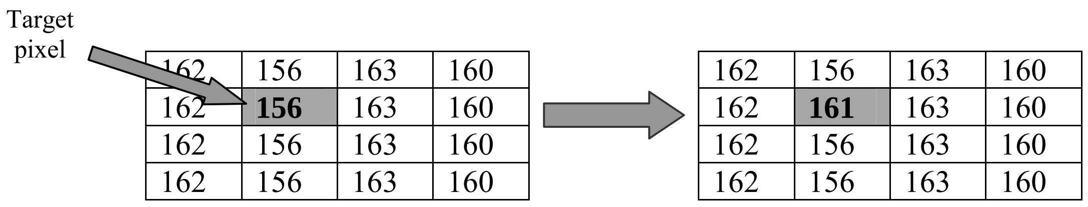
162	156	163	160
162	**156**	163	160
162	156	163	160
162	156	163	160

162	156	163	160
162	**160**	163	160
162	156	163	160
162	156	163	160

(a) Before mutation (b) After mutation

Figure (3): Example for mutation method

- Mutation 2 : In this method, when mutation occurs decided by it's relevant probability, a target pixel is chosen at random, and then sort the values of the block. The mutation operates by replacing the selected target value with the average value of the two middle values in the block after sorting as in figure below:



162	156	163	160
162	**156**	163	160
162	156	163	160
162	156	163	160

(a) Before mutation

162	156	163	160
162	**161**	163	160
162	156	163	160
162	156	163	160

(b) After mutation

Figure (4) : Example for mutation2 method

11- Offspring evaluation: The new offspring (s) evaluated by using the fitness-calculation algorithm, where each new offspring will have a new fitness value.

12- The replacement: It replaces the old individual that has tower fitness with new individual that has a better fitness. We used a Triple Tournament Replacement (TTR) in this paper by selecting three different individual randomly from the population, the new offspring will be replaced with the worst one of the three selected individual, if it's fitness larger than the worst one, otherwise, the new offspring do not replaced (Goldberg, 1989; Schmidt and Stidesen, 1997).

13- Termination Criterion: we used the on-line performance (Goldberg, 1989) as a termination criterion to measure the convergence of SSGA. The following algorithm explains the termination criterion.

```
Algorithm Termination-Criterion
  New-Avg ← 0
  Generation ← 1
  Flag ← 0
  Sum-Fit ← 0
  Do while ( Generation < MaxGen ) And ( Flag < Max-count )
     Old-Avg ← New-Avg
     ( Selection , Crossover, Mutation, Evaluation, Replacement )
     Sum-Fit ← Sum-Fit + Child.Fit
     New-Avg ← Sum-Fit / Generation
     If | New-Avg – Old-Avg | <  Difference  Then
         Flag ← Flag + 1
     Else    Flag ← 0
     End If
     Generation ← Generation + 1
  Loop
End of Termination-Criterion
```

Where New-Avg : The current average fitness
Old-Avg : The old average fitness
Difference : The permitted difference (difference =0.01)
Child . fit : new child fitness

Max-Count: the maximum number for difference in fitness (Max - count = 5)
Sum-Fit: Fitness sum
Generation: The current generation number.

14- Select the best solution: After the termination criterion is reached, we will select the best chromosome (block) that has best fitness as a best representative for it's unit in the codebook..

15- Put the best block in the final codebook that will used by either the encoder and decoder.

16- Increase the counter index by 1.

17- If the index less than N go to step 6 , otherwise the final codebook is filled , with the best representative blocks of their units.

18- Save the codebook at the file and give it to both encoder and decoder.

19- Reconstruct the image from codebook: The encoder will use the final codebook and the keys to reconstruct the received compressed image.

20- Evaluate the reconstructed image: The reconstructed image is evaluated by using the PSNR metric to measure the quality of the reconstructed image.

4. Simulation results:

In this section, we present computer simulation results to evaluate the performance of the proposed SSGA. We carried out extensive experiments to access the proposed SSGA for image compression. The SSGA is trained with varying parameters until it's PSNR value is optimal. The training vectors that used were 4×4 pixel at an 8-bits depth. Vector quantization output fixed-length codes for each tile. For 256 codeword codebook, the code length is 8 bits/tile. The image that used in this study are 512×512 pixel with 256 gray and color images. The SSGA used 65 chromosomes as a **popsize** and the maximum number of generation (**maxgen**) is 500. The image quality is measured by the PSNR that defined as follow:

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right)......................(7)$$

$$MSE = \frac{1}{k \times k}\sum_{i=1}^{p}\sum_{j=1}^{p}\left(X_{i,j} - Y_{i,j}\right)^2............(8)$$

Where $k \times k$ is the total number of pixels in the image , $X_{i,j}$ and $Y_{i,j}$ represent (i, j) pixel values in the original and reconstructed images respectively. We can measure the performance by using the following relations (Salmon, 2007):

1- $compression\ \text{ratio} = \frac{\text{size of output stream}}{size\ \text{of input stream}}..................(9)$

2- $100 * (1 - compreesion \text{ ratio})$.......................................(10)

3- $Bit \text{ rate} = \dfrac{\log_2 k}{m \times m}$...(11)

Where k is the number of codewords, $m \times m$ the block size.
The proposed system will produce a compression ratio using relation (9) equal to 0.015625 and by using the formula (10), it will produce 98.4375%. The system will produce 0.5 bpp using formula (11). We carried out many experiments to assess the proposed SSGA for image compression such as:

(a) study the effect of p_c and p_m on the image quality:

In this study, we will determine the accurate values for p_c and the p_m of the proposed SSGA. We will select the values of p_c and the p_m that maximized the PSNR of the reproduce image. The proposed SSGA used BTS, 2x, mutation1, and Lenna image in this study. Table (1) explain the effect of p_c on the quality of the reconstructed image with p_m=0.01. Table (2) shows the effect of the p_m on image quality.

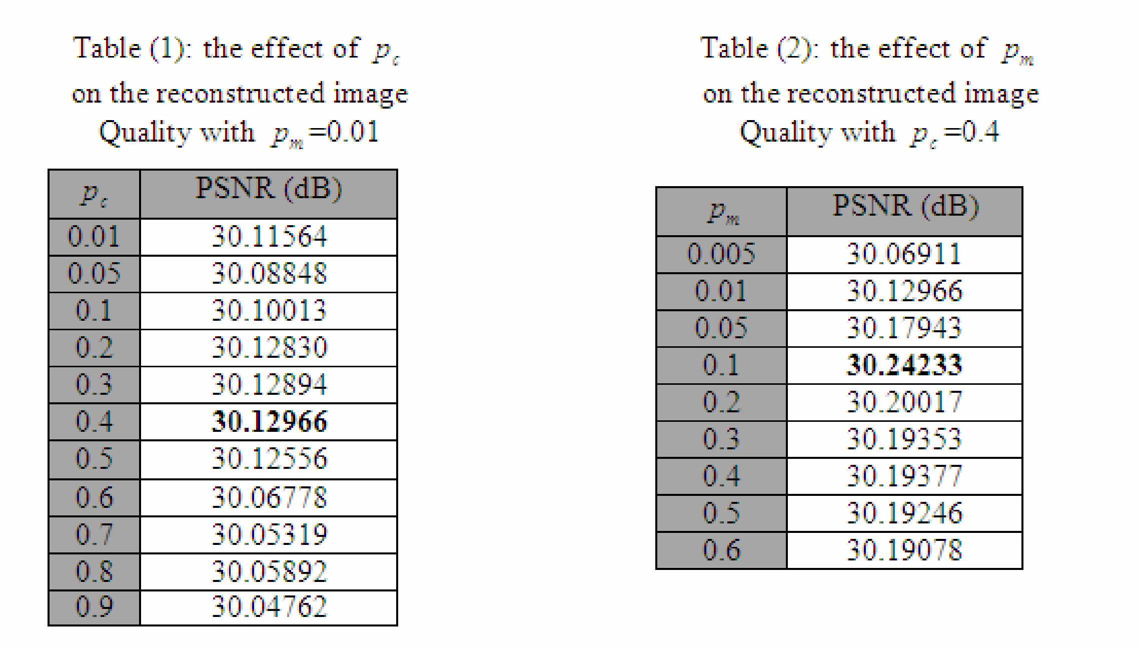


Table (1): the effect of p_c on the reconstructed image Quality with p_m=0.01

p_c	PSNR (dB)
0.01	30.11564
0.05	30.08848
0.1	30.10013
0.2	30.12830
0.3	30.12894
0.4	**30.12966**
0.5	30.12556
0.6	30.06778
0.7	30.05319
0.8	30.05892
0.9	30.04762

Table (2): the effect of p_m on the reconstructed image Quality with p_c=0.4

p_m	PSNR (dB)
0.005	30.06911
0.01	30.12966
0.05	30.17943
0.1	**30.24233**
0.2	30.20017
0.3	30.19353
0.4	30.19377
0.5	30.19246
0.6	30.19078

From the above simulation results, we see that the proposed SSGA converged to near global optimal solution with p_c= 0.4 and p_m=0.1 that maximize the PSNR of the reconstructed image.

(b) study the effect of the selection, crossover, and mutation methods on the quality of reconstructed image:

In this experiment, the proposed SSGA is trained with varying selection, crossover, and mutation methods to select the efficient methods that give a better PSNR value for Lenna image, where the selected methods will be used by the proposed SSGA with $p_c = 0.4$ and

p_m=0.1 . Table (3) explains the effect of selection, crossover, and mutation methods on the quality of image.

Table (3): the effect of selection, crossover, and mutation methods on the quality of image.

		Crossover Methods							
		1X	2X	UX	M1X	Arithmetic	Blend	Direction-Based	Sphere
BIS	Mutation1	30.113	30.138	30.128	30.151	30.209	30.232	30.241	30.207
	Mutation2	30.115	30.108	30.112	30.118	30.102	30.117	30.115	30.092
BTS	Mutation1	30.237	30.365	30.365	30.369	30.348	**30.373**	30.361	30.367
	Mutation2	30.073	30.173	30.164	30.168	30.157	30.174	30.163	30.168
HS	Mutation1	30.369	30.363	30.358	30.368	30.346	30.363	30.365	30.354
	Mutation2	30.202	30.185	30.254	30.257	30.119	30.163	30.160	30.174

From simulation results, we see that the proposed SSGA with BTS, Blend crossover, and mutation1 give larger PSNR value and a better visual quality compared with other methods.

(c) Test the proposed SSGA:

In this experiment, the proposed SSGA is tested on many standard images with size 512×512 pixel, where the SSGA used BTS, Blend crossover, and mutation1 with $p_c = 0.4$ and p_m=0.1. Table (4) explains the performance of the SSGA on many standard gray-scale images. Table (5) explains the performance of SSGA on color images. Where we used important measurement for compressed image quality is the PSNR value for all images.

Table (4): the performance of SSGA on gray-scale images

Image Name	PSNR(dB)
Lenna	30.37268
House	33.67423
Zelda	33.02779
Baboon	23.26847
Barbara	26.34609
Peppers	29.54647
Boat	27.85702
Goldhill	28.95977
Airplane f16	28.64048
Couple	26.84499
Aerial	24.90853
Splash	32.03002

Table (5): the performance of SSGA on color images

Image Name	PSNR(dB)
Lenna	22.99763
House	26.85416
Airplane f16	24.49097
Splash	26.02914
Jelly peans1	30.32648
Jelly peans2	29.13065
Peppers	24.17424
Tree	26.65270
Artichare	26.91878

From simulation results, we see that the proposed SSGA give a very good PSNR values and visual quality for the reconstructed gray-scale images, while it give a good PSNR values and visual quality for the reconstructed color images. We shows some of the original and reproduce gray-scale and color images that used by table (4) and table (5) in the following figures.

(a) (b)

Figure(5): Lenna image (a) original image. (b) reconstructed image

(a) (b)

Figure(6): Zelda image (a) original image. (b) reconstructed image

(a) (b)

Figure(7): Lenna image (a) original image. (b) reconstructed image.

(a) (b)

Figure(8): Airplane F16 image (a) original image. (b) reconstructed image.

5- The Conclusion and Future work:

In this paper, a SSGA based two phase image quantization algorithm for image compression is proposed. The initial codebook that used to categorize the image blocks by is created in the first phase. In second phase, the SSGA used to produce the best representative block for each unit in the codebook. We study many selection, crossover, and mutation methods and we find that the BTS, Blend crossover, and Mutation1 methods gives best results compared with other methods. The suggested fitness calculation algorithm participates in increasing the efficiency of the SSGA in generating the best representative block for each unit in the codebook. The proposed SSGA gave a very good PSNR values and visual quality for the reconstructed gray level images and it gave a good PSNR and visual quality for reconstructed color images. It produces a good compression ratio for both types of images. Future work includes developing a method that produces a best representative codebook for the entire image with larger PSNR value and high visual quality for reproduced image.

6. References

Armstrong, A. and Jiang, J.(2001). Genetic algorithms for image compression, Proceedings of IASTED International Conference Computer Graphics & Imaging, Honolulu, USA, pp179-184, ISBN: 0-88986-303-2.

Chambers, L. D. (2001). the practical handbook of genetic algorithms, applications, 2nd edition, Chapman& hall /CRC.

Cosman, P. C. , Gray, R. M. and Vetterli, M. (1996). Vector quantization of image subbands: a survey, Trans. IEEE image proc., vol. 5, No.2, pp.202-225.

Deb, K. (1998). Genetic algorithm in search and optimization: the technique and applications, proceeding of international workshop on soft computing and intelligent system, Calcutta, 58.

Delport, V. and Koschorreck, M.(1995). Genetic algorithm for codebook design in vector quantization , electronics letters, vol.31, No.2, pp.84-85.

Fränti, P., Kirijarvi, J., Kaukoranta, T. and Nevalainen, O. (1997). Genetic algorithms for codebook generation in vector quantization, Proceeding of the 3rd Nordic workshop on genetic algorithms (3NWGA), Helsinki, Finland, pp. 207-222.

Gersho, A. and Gray, R. M. (1992). vector quantization and signal compression, kluwer Academic publishers, Boston, Massachusetts.

Gen, M. and Cheng, R. (2000). Genetic algorithms and engineering optimization, John Wiley & Sons, Inc.

Goldberg, D. E (1989). Genetic algorithms in search, optimization and machine learning, Addison-Wesley.

Hourani, M.(2004). Genetic algorithm for continuous variable optimization with applications to queuing network and gene-clustering problems, master thesis, Miami university, Oxford, Ohio.

Lu, Z., Sun, S. and Zhang, Z.(1999). Vector quantization based on genetic algorithm, the 3rd International symposium on test and measurement (IST M_i-99), Xijan, china, unne,2-4, pp.285-290.

Ma, C. K. and Chan, C. K. (1991). Maximum descent method for image vector quantization , Electronic Letters, vol. 27, No. 12, pp. 1772-1773.

Merlo, G., Garam, F., Fernandez, V., Britos, P., Rossi, B. and Garacia-Martinez R.(1999). genetic-algorithm based image compression, proceedings del IV simposio Brasileiro de Automacao Inteligente, Paginas (en prensa). Escola Politecnica da universidade stadual dosao Paulo, Sanpablo, Brasil, Sep. 1999.

Michalewicz, Z. (1996). Genetic algorithms +Data structures= Evolutionary programs, third edition, Springer-Verlag.

Ng, W., Choi, S. and Ravishankar, C. V. (1995). An evolutionary approach to vector quantizer design, proceeding of the 2nd IEEE international conference on evolutionary computing (ICEC 95), Perth, Western Australia, pp. 406-411, Nov. 29-Dec.1.

Pan, J. S. , Mcinnes, F. R. and Jack, M. A. (1995). VQ codebook design using genetic algorithms, electronics leteters, Vol.31, No.17, PP. 1418-1419.

Po, L. M.(1990). Novel subspace distortion measurement for efficient implementation of image vector quantization, electronics Letters, vol.26, No. 29, pp. 480-482.

Ra, S. W. and Kim, J. K. (1993). A fast mean-distance ordered partial codebook search algorithm for image vector quantization, IEEE Trans. Circuits syst. II, vol. 40, pp. 576-579.

Salmon, D. (2007). Data Compression: The complete reference, 3rd edition, Springer-Verlag new York, Inc.

Schmidt, M. and Stidesen, T. (1997). Hybrid systems: genetic algorithms, neural networks and fuzzy logic, DAIMIIR.

Torres, L. and Huguet, J.(1994). An improvement on codebook search for vector quantization, IEEE Trans. Communication, com-42, pp. 208-210.