Comparative Study on VQ with Simple GA and Ordain GA

SADAF SAJJAD
COMSATS Institute of Information Technology
Department of Computer Science
Tobe Camp, Abbotabad
PAKISTAN

SAJJAD MOHSIN
COMSATS Institute of Information Technology
Department of Computer Science
Tobe Camp, Abbotabad
PAKISTAN

Abstract: In the present research we study the codebook generation problem of vector quantization, using two different techniques of Genetic Algorithm (GA). We compared the Simple GA (SGA) method and Ordain GA (OGA) method in vector quantization. SGA with roulette and tournament selection with elitist approach is used in the experiments. The OGA is based on the pair wise nearest neighbour method (PNN). Both these approaches were fine tuned by the inclusion of GLA. The two methods are campared with respect to quality of compressed image, rate of distortion and time cost. While using OGA we got better value of PSNR (34.6) with less distorted image as compared to the SGA with (29.7) PSNR value. Although in OGA the time performance is inferior, it is 3 times slower.

Key–Words: Genetic Algorithms, Vector Quantization, Codebook, Image Compression

1 Introduction

Image compression has been an important research topic for many years. There are many kinds of compression techniques. Lots of research literature is available which explains the importance and techniques of Image compression. Recently this field has gained an immense attention of scientists and researchers[1].

In 1980, Linde, Buzo, and Gray proposed a VQ design algorithm based on a training sequence. The use of a training sequence bypasses the need for multi-dimensional integration. A VQ that is designed using this algorithm are referred to in the literature as Generalized Lloyd Algorithm (GLA)[2]. Recently Vector Quantization is considered to be a most popular technique for image Compression [3]. GA has been successfully applied to codebook design for vector quantization. It tries to solve the given problem of image compression in an efficient way[$4-6$].

GA began with the work of Friedburg in the 1950's who mutated small Fortran programs to induce learning. John Holland reinvigorated the field in the mid-1970's by using bit strings as its representation and using the reproduction, crossover, and mutation operators. In the nineties, John Koza focused on improving the underlying representation language being used[7].

The main objective of this study is to generate and compare the codebook for a vector quantizer using SGA and OGA. The aim is to find a M code vectors for a given set of N training vectors using both the techniques. We than compare the two methods of GAs that are SGA and OGA on the vector quantization of images with respect to quality of compressed image, rate of distortion and time cost.

This paper is organized as follows section 1 is the introduction outlining the background and purpose of this study. In section 2 Codebook generation along with the experimental strategies for SGA and OGA are explained. Section 3 presents a discussion on comparisons of SGA and OGA with respect to the results. Finally the conclusion of the research is given in Section 4.

2 Codebook generation using Vector Quantization

We study the problem of generating a codebook for a vector quantizer (VQ). The aim is to find M code vectors (codebook) for a given set of N training vectors (training set).

An image is first converted into the set of $X = x_1, x_2, ..., x_N$ of N training vectors in a K-dimensional Euclidean space to find a codebook

$C = c_1, c_2, ..., c_M$ of M code vectors. The aim is to minimize the average distance between the training vectors and their representative code vectors. The distance between two vectors is defined by their squared Euclidean distance.

$$d^2 = \sum_{i=1}^{N} ||x_i - c_{pi}||^2. \qquad (1)$$

The distortion of the codebook is then calculated as

$$D(P,C) = \frac{1}{N} d^2. \qquad (2)$$

For codebook generation, two optimizations are essential, the first is the Partition optimization and the second is the codebook centroid optimization.

The optimization of partition P is obtained by placing each training vector x_i to its nearest code vector c_j so as to minimize euclidian distance (1) that in turn optimize (3) that can be given by

$$p_i = \arg min ||x_i - c_j||^2. \qquad (3)$$

The optimization of the codebook C is obtained by calculating the centroid of the clusters c_j as the code vectors

$$c_j = \frac{\sum_{p_i=j} x_i}{\sum_{p_i=j} 1}, 1 \leq j \leq M \qquad (4)$$

that in turn optimizes (3).

Conventional PNN by Equitz [8] uses the hierarchical approach of generating codebook. It starts by considering each vector as a separate code vector. Then it converge/merge the two vectors whose distortion is minimum. This process goes on till the desired size of the codebook is achieved. The distortion of the merge is calculated as

$$d_{a,b} = \frac{n_a n_b}{n_a + n_b} . ||c_a - c_b||^2 \qquad (5)$$

where c_a and c_b are the merged code vectors, n_a and n_b are the size of the corresponding clusters. The PNN approach is used in OGA.

Franti and Kaukoranta (1998) introduced a faster method of PNN [6]. Their main idea is to maintain the pointer of nearest neighbour to avoid the calculation of the distance between the two neighbours. After each merge the pointer table is to be updated for the merged vectors only. This in turn reduces computational time significantly. The most cited and widely used method is GLA [2]. It starts with an initial codebook, which is iteratively improved until a local minimum is reached. The result of the GLA is highly dependent on the choice of the initial codebook. Better results can be obtained by using an optimization technique known as GA [7].

2.1 Simple GA

SGA starts with initial random selection of population. In our case the fitness function is calculated by calculating the distortion through (Eq. 2).

All the chromosomes are encoded into the binary strings and genes were selected for crossover through various selection criteria in order to get better results. Roulette selection, tournament selection with elite gene and without elite is done. In roulette selection the individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness. A random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals are obtained.

In tournament selection 2 individuals are chosen randomly from the population and the best individual from this group is selected as parent. This process is repeated till the number of required parents are reached.

As Elitism can rapidly increase the performance of GA, and it prevents a loss of the best-found solution so we first copies the best chromosome to the new population and then the rest of the population is constructed again.

Mutation is performed as an exchange of two genes on the result of crossed over genes and the process again starts with the population selection for T times. The probability for cross over is 60% and for mutation it is 1% in our case. Fig. 1 shows the flow diagram and the operators of SGA.

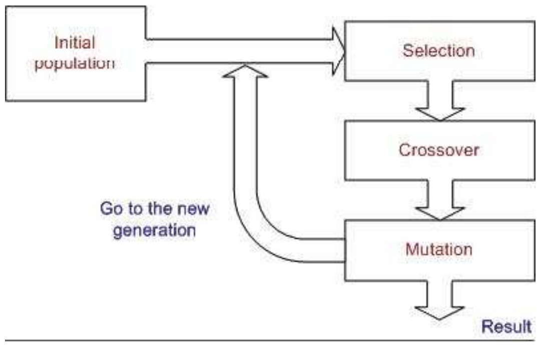

Figure 1: Illustrating the Simple GA

2.2 Ordain GA

A very basic and most important choice in the GA is the way solution is represented, since the data structure is determined through it. That in turn are modified through crossover and mutation to give optimal solution.

In 2000 Pasi Franti used the combination of partition and the codebook together to overcome the computational inefficiency of recalculating Partition optimality P and Codebook optimality C [5].

The selection method in both SGA and OGA is based on the Darwin theory of "survival of the fittest". In OGA, though selection is done in a greedy way in which all possible pairs are permutated. Than they are organized in an increasing order by their distortion values. With elitist approach solution with least distortion are selected and other are discarded.

In OGA the crossover method is different from the SGA. In this method the two existing solution in order are combined to form a new solution. In addition to that, unnecessary computation is not wasted for a complete repartition but the partitions of the parent solutions are utilized.

It starts by taking the union of the two existing solution (i.e. parents) in order, and than merging them. The partition closer to the vector (smaller in distance) is chosen. The new codebook is updated using (4) in respect to the new partition. In this way we get twice the size of codebook (i.e. $2M$ instead of M).

The final size of the codebook is then obtained using the PNN algorithm. The first step of the PNN is to search for each code vector its nearest neighbour that minimizes the merge cost according to (5). After the merge, the pointers are updated, and the process is repeated until the size of the codebook is reduced to M.

In Mutations we randomly chose code vector first, then we randomly chose training vector and than we swap them. This method is denoted as random swap. Its basic purpose is to discover new search paths when the population becomes too homogenous for the crossover to achieve significant improvement anymore. Mutations if becomes vital, suggests that the crossover is not well-defined and needed to be modified.

3 Results and Discussions

The problem of generating a codebook for a vector quantizer (VQ) is studied. For a given set of N training vectors we find M code vectors. To evaluate the difference between simple and ordain GA, both algorithms were compared here for image compression. They are both coded in Matlab language and run on UNIX. We consider a set $X = x_1, x_2, ..., x_N$ of N training vectors to find a codebook $C = c_1, c_2, ..., c_M$ of M code vectors. The distortion of the encoded picture is measured by the peak-signal-to-noise ratio (PSNR). In order to perform comparative testing, number of generations was kept constant for all the methods of GA.

3.1 Training Sets and Statistics

Two images, "Lena" and "Bridge" with resolution 256x256 pixels, 8 bits per pixel, are used here. The images are divided into 4x4 blocks for training; the codebook size is $M = 256$.

Image	Bits per pixel	Number of Vectors
LENA	8	4096
BRIDGE	8	4096

Table 1: Training Sets and Statistics

Training Set Images are shown in Fig. 2

Figure 2: Lena, Bridge (256x256)

3.2 Computational Comparison Performance for (PSNR)

The PSNR values are given in table 2. The results show that the OGA performs better in terms of PSNR over SGA. The fine tunning through GLA has improved the PSNR values for both GA.

where RW stands for Roulette Wheel and T stands for Tournament selection.

After we apply the series of experiments for SGA and OGA we get the images in compressed forms. The images Figures [3 – 7].

Image	SGA RW	SGA T + elite	SGA + GLA	OGA	OGA + GLA
LENA	25.9	33.6	29.7	34.5	34.6
BRIDGE	26.0	31.5	29.2	32.1	32.1

Table 2: Comparison Performance (PSNR) of Various Methods of GA

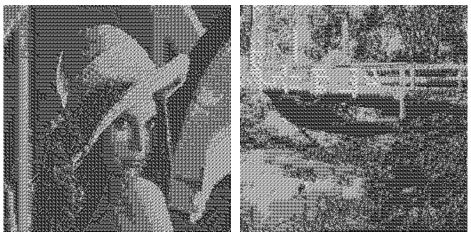
Figure 3: GA with roulette: Lena, Bridge

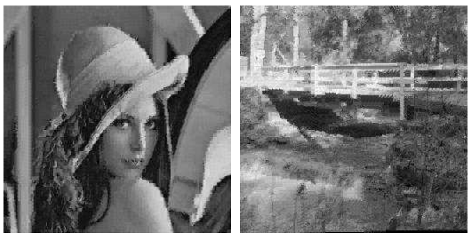
Figure 4: GA with tournament+elite: Lena, Bridge

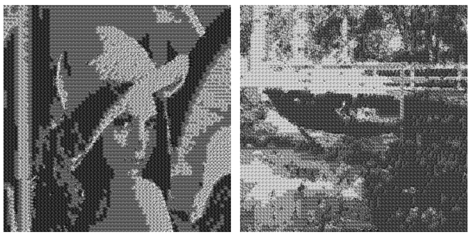
Figure 5: GA with GLA: Lena, Bridge

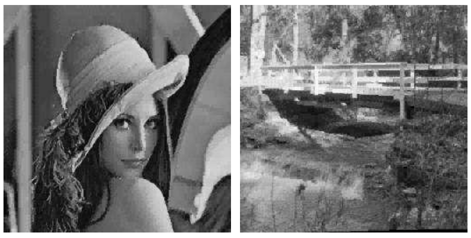
Figure 6: Ordain GA: Lena, Bridge

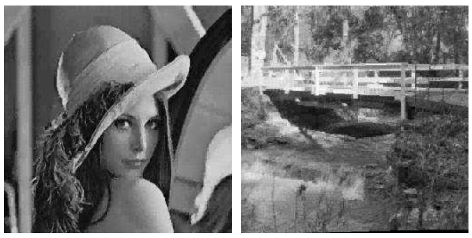
Figure 7: Ordain GA with GLA: Lena, Bridge

3.3 Computational Comparison Performance for Time

The time taken for both the algorithms were compared. It is noted that simple GA consumes less time as compared to ordain GA. It is also found that inclusion of GLA does not effect the overall time for both. The results are shown in table 3.

Image	SGA RW	SGA T + elite	OGA	OGA + GLA
LENA	61:19	42:01	117:28	117:30
BRIDGE	61:01	42:01	118:00	118:00

Table 3: Comparison Performance (TIME) of Various Methods of GA

3.4 Comparative Discussion

Results show that OGA gives high Peak signal noise ratio value and less distortion as compared to the SGA. This is because in OGA we are considering all X vectors as the solution space. Whereas in SGA we have randomly selected M number of vectors from input vectors X as a sample and performing crossover and mutation among these M number of vectors. The optimal solution can be outside the sample space.

Time taken to compress the image in SGA is less because it is computing between only M number of vectors while OGA is computing X number of vectors (where M ¡ X).

After our experiments we found three main differences in the results of SGA and OGA, which are described in table 4.

4 Conclusion

In this research, the problems of codebook design are studied and vector quantization with two methods of GA is performed. These two GAs are

1. SGA takes M number of vectors randomly from an input space of X vectors to get optimal solution	1. OGA takes all the X vectors from an input space for finding optimal solution
2. Values in the vectors are selected for crossover and mutation in binary strings.	2. Vectors as a whole are crossed over and mutated.
3. Computational time decreases because it works on M number of randomly selected vectors from input vectors X.	3. Computational time increases because it works on X number of input vectors.

Table 4: Comparison Discussion

Simple Genetic Algorithm with the use of different selection criteria and Ordain Genetic Algorithm with the pairwise nearest neighbour, results are also fine tuned for both the algorithms with Generalized Lloyd algorithm. In order to compare the results, experiments are performed under the same environment and number of runs for every algorithm is kept 100.

It is concluded from the results that Simple GA needs shorter computational time than Ordain GA but ordain GA can generate a codebook of high quality with the least distortion and more PSNR value.

References:

[1] R. C. Gonzelez, and R. E. Woods, "Digital Image Processing", Newyork, Addison-Wesley, 1993.

[2] Y. Linde, A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizater Design", IEEE Transaction on Comm., pp. 84-95 Vol. 28 January 1980.

[3] A. Gresho and R. Gray, " Vector Quantization and Signal Compression" Bostan MA: Kluwer, 1992.

[4] P. Franti, J. Kivijarvi, T. Kaukoranta and O. Nevalainen, "Genetic algorithms for large scale clustering problem", The Computer Journal, 40 (9), 547-554, 1997.

[5] P. Franti, "Genetic algorithm with deterministic crossover for vector quantization", Pattern Recognition Letters, 21 (1), 61-68, 2000.

[6] P. Franti, T. Kaukoranta, D.-F. Shen and K.-S. Chang, "Fast and memory efficient implementation of the exact PNN", IEEE Transactions on Image Processing, 9 (5), 773-777, May 2000.

[7] M.Mitchell, "An Introduction To Genetic Algorithms", MIT press, Cambridg,e England, (1998).

[8] W.H. Equitz, "A new vector quantization clustering algorithm", IEEE Trans. on Acoustics, Speech and Signal Processing, pp. 1568-1575 Vol. 37, 1989.