This paper appeared in the Second IEEE International Conference on Evolutionary Computing, Perth, Western Australia, November 29–December 1, 1995

An Evolutionary Approach to Vector Quantizer Design*

Wee K. Ng† Sunghyun Choi Chinya V. Ravishankar

Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109-2122
E-mail: {wkn,shchoi,ravi}@eecs.umich.edu

Abstract

Vector quantization is a lossy coding technique for encoding a set of vectors from different sources such as image and speech. 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 known to be difficult [3]. The conventional solution technique works through a process of iterative refinements which yield only locally optimal results. In this paper, we design and evaluate three versions of genetic algorithms for computing vector quantizers. Our preliminary study with Gaussian-Markov sources showed that the genetic approach outperforms the conventional technique in most cases.

1 Introduction

Vector Quantization (VQ) is a lossy source coding technique that maps a sequence of continous or discrete k-dimensional vectors into a digital sequence suitable for communication over or storage in a digital channel [2, 5, 11]. The goal is data compression: to reduce the bit rate so as to minimize communication channel capacity or digital memory requirements while maintaining the necessary fidelity of the data.

Given an input sequence vectors, VQ attemts to construct a set of representative vectors (*codewords*) called the *codebook* such that each input vector may be replaced by a vector from the codebook. As the input vector and its representative vector are not often identical, a *distortion* is introduced. The problem of VQ design is to find a codebook that has the least overall distortion for a given set of input vectors. The problem may be formally defined as follows:

> *Given a set X of m k-vectors, find a codebook Y of n k-vectors such that the total distortion $\sum_{i=1}^{m} d(\bar{x}_i, Q(\bar{x}_i))$, $Q(\bar{x}_i) \in Y$, is minimized.*

This problem is known to be difficult and there are no known closed-form solutions [3].

The conventional technique for designing a codebook works through a process of *iterative refinements* of an initial codebook [9]. A brief description is given in Section 2. This technique does not gaurantee optimality, it sometimes yield locally optimal codebooks. Genetic algorithms [4, 7] are becoming a widely used and accepted method for very difficult problems [1]. There has been some recent work in the application of genetic algorithms to the area of coding. In [12], genetic algorithms are used to find an optimum dictionary for lossless coding while in [10], they are used for linear block decoding. In this paper, we present a genetic algorithm to approximate a solution to the optimal codebook design problem. To the best of our knowledge, this is the first application of its kind to the domain of vector quantization.

The organization of this paper is as follows. In Section 2, we describe the conventional algorithm for computing codebooks. Section 3 describes the configuration of our genetic algorithm. We perform some simulations and compare the performance between the genetic and conventional algorithms in Section 4. Finally, the last section concludes with a summary of our work and directions for future research.

2 Generalized Lloyd Algorithm

A widely used technique for codebook design is the Generalized Lloyd Algorithm (GLA) [9] as shown in Figure 2. The algorithm begins with a set of input vectors and an initial codebook. For each input vector, a codeword from the codebook is chosen that yields the minimum distortion. If the sum of distortions from all input vectors does not improve beyond some threshold, the algorithm stops. Otherwise the codebook is

*This work was supported in part by the Consortium for International Earth Science Information Networking.

†The author may also be reached at his World Wide Web home page at URL: "http://www.eecs.umich.edu/~wkn".

Step 0	Given: An input sequence X and an initial codebook Y.
Step 1	Map each vector $\bar{x} \in X$ into a vector in Y that yields the minimum distortion.
Step 2	Compute total distortion for all $\bar{x} \in X$.
Step 3	If total distortion does not improve beyond a threshold, quit.
Step 4	Update the codebook Y as follows: Replace each codeword $\bar{y} \in Y$ by the centroid of all input vectors that are mapped into it in Step 1.
Step 5	Go to Step 1.

modified as follows: Each codeword is replaced by the *centroid* of all input vectors that have previously chosen it as their output vector. This completes one iteration. The GLA iteratively refines a codebook.

GLA is a *decent* algorithm; each iteration reduces the average distortion and corresponds to a *local* change in the codebook, i.e., the new codebook is not drastically different from the old codebook. Given an initial codebook, the algorithm leads to the *nearest local* codebook, which may not be optimal. As codebook design is a complex optimization problem, it has many local minima. Thus, the chance of GLA pinpointing the optimal codebook is slim.

3 Genetic Algorithm Design

We implemented three versions of genetic algorithms for the codebook design problem based on GLA. They are collectively called the Genetic Generalized Lloyd Algorithms (GGLA) and the three versions are referred to as GGLA-I, GGLA-II and GGLA-III. We first describe features that are common to all three; namely chromosome representation, genetic operators and fitness value definition:

- **Chromosome representation:** Each chromosome in our genetic algorithm represents a codebook. The set of genes in a chromosome corresponds to the codewords. A population consists of a set of codebooks.

- **Fitness value:** The fitness value of a chromosome is the *signal-to-noise* ratio (SNR) defined as follows: Given a set X of m k-vectors and a codebook Y of n k-vectors, the average distortion per symbol $D = 1/(km)\sum_{i=1}^{m} \|\bar{x}_i - Q(\bar{x}_i)\|^2$ where $Q(\bar{x}_i) \in Y$ is the codeword for $\bar{x}_i$. Then, SNR $= 10\log_{10}(\sigma^2/D)$ where variance $\sigma^2 = 1/(km)\sum_{i=1}^{m}\sum_{j=1}^{k}(x_{i,j} - \mu)^2$ and mean $\mu = 1/(km)\sum_{i=1}^{m}\sum_{j=1}^{k} x_{i,j}$.

- **Genetic operators:** The *mutation* operator modifies a gene (vector) by multiplying its elements by a random number between 0.8 to 1.25 with probability 0.001. The *crossover* operator exchanges segments of two chromosomes. The size of the segment is determined randomly. We also adopt a chromosome replacement strategy by replacing low fitting chromosomes with high fitting chromosomes. If the SNR of the chromosome with the lowest SNR is lower than the average SNR of all chromosomes less one, it will be replaced by the chromosome with the highest SNR. This process is repeated with subsequent pairs of chromosomes containing the next lowest and highest SNRs. The number of replacements is limited to no more than one third the size of the population.

The three versions of GGLA and their differences are described below. All three versions perform chromosome replacements at the beginning of each iteration.

- **GGLA-I:** Chromosomes are randomly paired up. If the two chromosomes in any pair show no further SNR improvements, they are crossovered. Otherwise, the GLA is used to refine the chromosomes.

- **GGLA-II:** Any two chromosomes that show no further improvements are crossovered. Otherwise, they are refined by GLA. This differs from GGLA-I in that the chromosome are not paired up initially.

- **GGLA-III:** Chromosomes are randomly paired up. Each pair of chromosome is then crossovered or simply refined by the GLA with probabilities p and $(1-p)$ respectively.

In all three versions, the mutation operator is always applied at the end of the iteration.

α	Vector dimension	Codebook	Max SNR (dB)
0	4	4	2.079
		8	3.558
		16	4.967
		32	6.531
		64	8.314
	2	16	9.845
		32	12.691
		64	15.729
		128	18.824
		256	22.050
0.5	4	4	3.209
		8	4.609
		16	6.057
		32	7.591
		64	9.298
	2	16	10.442
		32	13.245
		64	16.196
		128	19.362
		256	22.726
0.9	4	4	6.946
		8	8.836
		16	10.645
		32	12.330
		64	14.010
	2	16	13.625
		32	16.290
		64	19.184
		128	22.167
		256	25.507

Table 1: GLA performance for first-order Gaussian-Markov source.

4 Performance Evaluation

We perform numerous simulations on the three versions of GGLAs using samples generated by first-order Gaussian-Markov processes. The SNRs obtained are compared with those achieved by the GLA.

4.1 Gaussian-Markov Source

Each first-order Gaussian-Markov sample x_i is generated using the equation: $x_i = \alpha x_{i-1} + w_i$ where α is a correlation coefficient and w_i is a sample of independent white Gaussian noise. We used three α values of 0.0, 0.5 and 0.9.

The generated samples are used to initialize the population of chromosomes as follows: The sequence of samples are first partitioned into blocks of k samples to form k-vectors. Then, n of these vectors are grouped to form a chromosome. The elements of the vectors are multiplied by a random number between 0.8 and 1.25 so that the vectors obtained are different. We generated two sets of samples. In the first set, there are 4096 samples blocked into 1024 4-dimensional input vectors. In the second set, there are 8192 samples blocked into 4096 2-dimensional input vectors.

4.2 Simulation Parameters

All simulations are performed with population size of 10 and 200 iterations. For the first sample set, the codebook sizes are $4, 8, 16, 32, 64$. In terms of *coding rate*, these correspond to rates of $R = 0.5 \sim 1.5$. ($R = \log_2 n/k$ where n is the codebook size and k is the vector dimension.) For the second sample set, the codebook sizes are $16, 21, 64, 128, 256$ which yield rates of $R = 2.0 \sim 4.0$. For each combination of codebook size, vector dimension and correlation coefficient α, GLA and the three GGLAs are executed.

4.3 Simulation Results

Table 1 shows the performance of the GLA. The performances of the three GGLAs are shown in Figure 1. The maximum SNR figures in Table 1 are used as a reference for comparison with the GGLA performances in Figure 1. We observe that in almost all cases, the GGLAs outperformed the GLA and the improvements are larger for higher rates. In particular, GGLA-III with $p = 0.7$ made a remarkably 0.6 dB improvements for $\alpha = 0.9$, $k = 2$ and $R = 4.0$.

Figure 2 shows another interpretation of the differences between GLA and the GGLAs. The parameters used are: $\alpha = 0.9$, $k = 4$, $n = 64$ and $R = 1.5$. Compared with the GGLAs which handle multiple codebooks at a time, the conventional algorithm operates on a single codebook only. To present a better comparison of the SNRs achieved, we made a modification to GLA so that when the SNR improvement of the codebook is less than 0.0001 (dB), GLA discards the codebook and starts with a new one. This explains the spikiness of the solid graph in Figure 2. In addition, the total number of initial codebooks used by GLA is approximately equal to the number of new chromosomes produced by each GGLA.

We observe that (1) the average SNR of GLA fluctuates even though GLA is a decent algorithm; (2) there are many occasions in GLA where the maximum SNR saturates and then drops abruptly only to rise again; and (3) the average and maximum SNRs achieved by GLA is less than those of GGLAs.

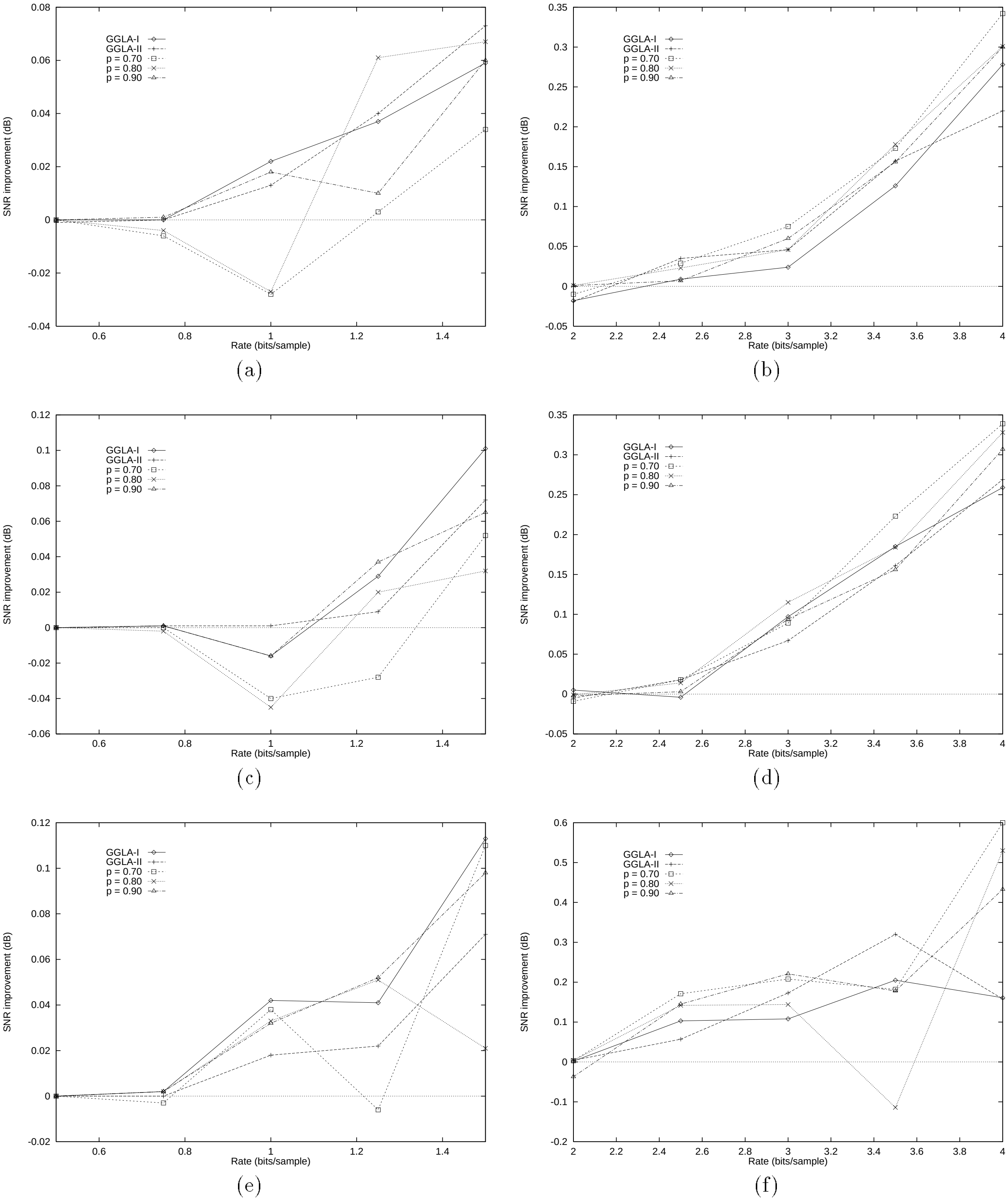
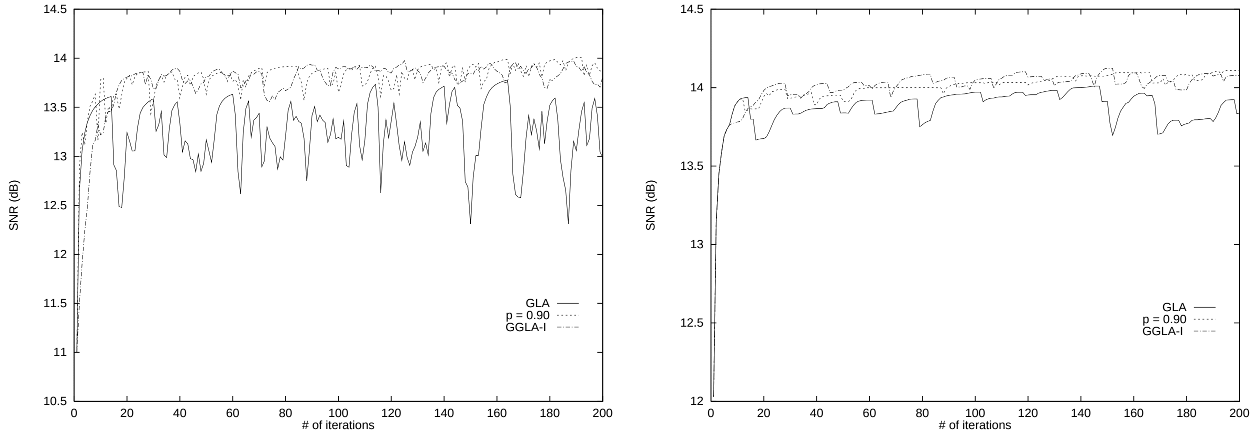


Figure 1: The SNR improvements obtained with GGLA-I, GGLA-II and GGLA-III. The simulation parameters are: (a) $\alpha = 0$, $k = 4$, (b) $\alpha = 0$, $k = 2$, (c) $\alpha = 0.5$, $k = 4$, (d) $\alpha = 0.5$, $k = 2$, (e) $\alpha = 0.9$, $k = 4$ and (f) $\alpha = 0.9$, $k = 2$. The graphs on the left and right correspond to the two sets of Gaussian-Markov samples.



(a) Average SNR

(b) Maximum SNR

Figure 2: Average and maximum SNR versus numbers of iterations. The parameters are: $\alpha = 0.9$, $k = 4$, $n = 64$ and $R = 1.5$.

4.4 Discussion

A brief conclusion we can make is that GGLA outperforms GLA in most cases. Let us examine intuitively why this is so. As noted earlier in Section 2, GLA makes small local changes at each iteration. Depending on the initial codebook, it may reach a local minimum where some of its codewords are not chosen by any of the input vectors for output. The introduction of various genetic operators permit the elimination of these codewords and introduce new codewords from another codebook. In other words, these genetic operators effect a global change that is impossible with the GLA.

5 Conclusions

We have examined a genetic algorithmic approach to the codebook design problem. To the best of our knowledge, this is the first application of its kind to vector quantizer design. Our study showed that in most cases, the genetic approach yields better solutions than the conventional algorithm. However, the cost of GGLA is more memory; we need as many times the amount of memory as the population size.

As this work is still preliminary, we conclude with some suggestions for future study. This includes (1) experimenting with other combinations of population sizes, codebook sizes, vector dimensions; (2) using other source samples such as image and speech, and (3) conducting a theoretical investigation of why GGLA outperforms GLA.

References

[1] K. A. DeJong, W. M. Spears. Using Genetic Algorithms to Solve NP-Complete Problems. *Proceedings of the 3rd International Conference on Genetic Algorithms*, pp. 124–132, Tuscaloosa, Alamaba, June 4–7, 1989.

[2] A. Gersho. On the Structure of Vector Quantizers. *IEEE Transactions on Information Theory*, Vol. 28, No. 2, pp. 157–166, March 1982.

[3] A. Gersho, R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, Massachusetts, 1992.

[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

[5] R. M. Gray. Vector Quantization. *IEEE ASSP Magazine*, Vol. 1, pp. 4–29, April 1984.

[6] R. M. Gray, J. C. Kieffer, Y. Linde. Locally Optimal Block Quantizer Design. *Information and Control*, Vol. 45, pp. 178–198, May 1980.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, 1992.

[8] Y. Linde, A. Buzo, R. M. Gray. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications*, Vol. 28, No. 1, pp. 84–95, January 1980.

[9] S. P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, Vol. 28, No. 2, pp. 127–135, March 1982.

[10] H. Maini, K. Mehrotra, C. Mohan, S. Ranka. Genetic Algorithms for Soft-Decision Decoding of Linear Block Codes. *Evolutionary Computation*, Vol. 2, No. 2, pp. 145–164, MIT Press, Cambridge, Massachusetts, Summer 1994.

[11] N. M. Nasrabadi, R. A. King. Image Coding Using Vector Quantization: A Review. *IEEE Transactions on Communications*, Vol. 36, pp. 957–971, August 1988.

[12] W. K. Ng, C. V. Ravishankar. A Preliminary Study of Genetic Data Compression. *Proceedings of the 6th International Conference on Genetic Algorithms*, L. Eshelman (ed.), Morgan Kaufmann, San Francisco, California, 1995.