VQ Codevector Index Assignment Using Genetic Algorithms for Noisy Channels

J. S. Pan†‡, F. R. McInnes‡, M. A. Jack‡
†Department of Electronic Engineering, Kaohsiung Institute of Technology, Taiwan
‡Centre for Communication Interface Research, University of Edinburgh, UK

ABSTRACT

The multi-mutation rates, multi-crossover rates and a scheme of reinitialization are applied to parallel genetic algorithm for assigning the codevector indices for noisy channels for the purpose of minimizing the distortion caused by bit errors. Experimental results based on the memoryless binary symmetric channel for any bit error demonstrate the robustness of this new approach compared with our previous work [1]. The property of multiple global optima is also emphasized in this paper.

1 INTRODUCTION

Vector quantization (VQ) [2] has received considerable attention due to the dramatic bit rate reduction. A vector $X = \{x^1, x^2, ..., x^k\}$ consisting of k samples of information source in the k-dimensional Euclidean space R^k is sent to the vector quantizer. The k-dimensional vector quantizer with the number of codevectors N is defined as follows by using the reproduction alphabet consisting of N codevectors, $C = \{C_1, C_2, ..., C_N\}$, the partitioned set consisting of subspaces of the k-dimensional Euclidean space R^k, $S = \{S_1, S_2, ..., S_N\}$, and the mapping function $Q(\cdot)$:

$$Q(X) = C_i, \qquad if \quad X \epsilon S_i. \tag{1}$$

The sets S_i satisfy

$$\cup_{i=1}^{N} S_i = R^k. \tag{2}$$

and

$$S_i \cap S_j = \phi \qquad if \quad i \neq j. \tag{3}$$

The output of the vector quantizer is the index i of the codevector C_i which satisfies

$$i = argmin_p \sum_{l=1}^{k} (x^l - c_p^l)^2. \tag{4}$$

Only the index i is transmitted over the channel to the receiver. The performance of vector quantizer can be evaluated by the squared Euclidean distortion per symbol given by

$$D_s = \frac{1}{k} \sum_{i=1}^{N} \int_{S_i} P(X) \sum_{l=1}^{k} (x^l - c_i^l)^2 dX, \tag{5}$$

where $P(X)$ is the probability density function of X.

The channel noise will induce channel errors in the communication. The effect of channel errors is to cause errors in the received indices. Thus, distortions are introduced in the decoding step. Distortion due to an imperfect channel can be reduced by assigning suitable indices to codevectors. If the number of codevectors is N, the possible combination of indices to codevectors is $N!$. To test $N!$ assignments is an NP-hard problem. Zeger and Gersho [3] proposed the binary switching algorithm to improve the codevector index assignment. Farvardin [4] applied the simulated annealing technique to design the codevector indices. Wu and Barba [5] developed an efficient index allocation algorithm by using the information of a priori probability of codevectors. In this paper, more experiments have been done for our previous proposed algorithm [1]. A new approach is presented by applying the different mutation rate and different crossover rate for different subpopulation and the scheme of reinitialization to the parallel genetic algorithm to enhance the robustness of the parallel genetic algorithm for codevector index assignment.

The motivation of the scheme of reinitialization is to avoid the early convergence in the poor solution. When most of the individuals have the same chromosomes, there is almost no chance to jump off the local optimum. At this state, the best individual can be used as a template to reinitialize the population by swapping the genes in the chromosome several times randomly. The mutation rate and crossover rate are important parameters in genetic algorithms. The values of mutation rate and crossover rate depend on the applications. In parallel genetic algorithm, the mutation rate and crossover rate can be set to different value for different subpopulation to enhance the function of crossover and mutation operators.

2 Average Distortion and Multiple Global Optima

N codevectors C_i, $i = 1, 2, ..., N$, are assigned codevector indices with an m bit string $b(c_i)$, where $N = 2^m$. The distortion between codevector C_i and C_j is given by a non-negative distortion measure $d(c_i, c_j)$. Usually, the Euclidean metric is used. Let $P(b(c_j)/b(c_i))$, $i, j = 1, 2, ...N$, denote the probability that the index $b(c_j)$ is received given the index

opt	000	001	010	011	100	101	110	111
$\oplus$ 000	000	001	010	011	100	101	110	111
$\oplus$ 001	001	000	011	010	101	100	111	110
$\oplus$ 010	010	011	000	001	110	111	100	101
$\oplus$ 011	011	010	001	000	111	110	101	100
$\oplus$ 100	100	101	110	111	000	001	010	011
$\oplus$ 101	101	100	111	110	001	000	011	010
$\oplus$ 110	110	111	100	101	010	011	000	001
$\oplus$ 111	111	110	101	100	011	010	001	000

Table 1: Example of 2^3 possibilities for q_{ij}, $j = 1, 2, 3$, $i = 1, 2, ..., 8$

pst	globally optimal indices							
321	000	100	010	110	001	101	011	111
231	000	010	100	110	001	011	101	111
312	000	100	001	101	010	110	011	111
132	000	010	001	011	100	110	101	111
213	000	001	100	101	010	011	110	111
123	000	001	010	011	100	101	110	111

Table 2: Example of 3! possibilities for the permutation of bit strings b=(000, 001, 010, 011, 100, 101, 110, 111)

$b(c_i)$ is sent. Assuming random assignment of the codevector indices $b = (b(c_1), b(c_2), ..., b(c_N))$, the average distortion for any possible bit errors caused by the channel noise is given by

$$D_c = \frac{1}{N!}\sum_{i=1}^{N} P(c_i)\sum_{b}\sum_{j=1}^{N} P(b(c_j)/b(c_i))d(c_i, c_j). \tag{6}$$

We assume that the channel is a memoryless binary symmetric channel with bit error probability ε. Thus, the error probability is $\varepsilon^l(1-\varepsilon)^{m-l}$, where l is the number of bits in which $b(c_i)$ and $b(c_j)$ differ. Let $H(b(c_i), b(c_j))$ denote the Hamming distance between $b(c_i)$ and $b(c_j)$. The average distortion can be derived [1] as

$$D_c = \frac{1-(1-\varepsilon)^m}{N-1}\sum_{i=1}^{N} P(c_i)\sum_{j=1}^{N} d(c_i, c_j) \tag{7}$$

After the indices are assigned to the codevectors, the expectation of distortion for the transmission of indices $b(c_i)$, $i = 1, 2, ..., N$, can be written as

$$D = \sum_{i=1}^{N} P(c_i)\sum_{l=1}^{m} \varepsilon^l(1-\varepsilon)^{m-l}\sum_{b(c_j)\epsilon N^l(b(c_i))} d(c_i, c_j) \tag{8}$$

where $N^l(b(c_i)) = \{b(c_j)\epsilon I, H(b(c_i), b(c_j)) = l\}$, is the lth neighbour set of $b(c_i)$. Assume $f(c_i) = b_i = (b_{i1}, b_{i2}, ..., b_{im})$ is the function of index assignment. Here $b_{ij}\epsilon\{0, 1\}$, $i = 1, 2, ..., N$, $j = 1, 2, ..., m$. If f is globally optimal, then so is g defined by $g(c_i) = (a_{i1}, a_{i2}, ..., a_{im})$, where $a_{ij} = b_{ip(j)} \oplus q_{p(j)}$, $q_j\epsilon\{0, 1\}$, p is a permutation of $\{1, 2, ..., m\}$. There are 2^m possibilities for q_j, $j = 1, 2, ..., m$, and $m!$ possibilities for p. Thus, at least $m!N$ global optima exist for the problem of codebook index assignment. So, an $N!$ search space can be reduced to an $\frac{(N-1)!}{m!}$ search space. If the number of codevectors is 8 and the globally optimal assignment of the codevector indices $b = (000, 001, 010, 011, 100, 101, 110, 111)$, then there are 8 possible combinations for q_{ij}, $j = 1, 2, 3$, i.e., $H(b(c_i), b(c_l)) = H(b(c_i) \oplus s, b(c_l) \oplus s)$, $i = 1, 2, ..., 8$, $l = 1, 2, ..., 8$ and $s\epsilon\{000, 001, 010, 011, 100, 101, 110, 111\}$ which is depicted in Table 1. There are also 6 possibilities of using a permutation in the bit string for each possible combination in Table 1. One example is shown in Table 2.

3 Parallel Genetic Algorithm

Genetic algorithms [6,7,8,9] are adaptive methods which can be used in search and optimization problems. Here, a parallel genetic algorithm with the scheme of reinitialization, the multi-mutation rates and multi-crossover rates is used to optimize the codevector index assignment. The $fitness$ is the inverse of the expectation of distortion as in Eq. 8. The $chromosome$ is the index string. The proposed algorithm consists of the following steps :

1. Initialization – Randomly assign the indices (i.e. 0 to $N-1$) to every individual of the population. A $chromosome$ is composed of N indices. Separate the population into G groups. G sets of P members are generated in this step, where P is the population size for each group. Without loss of generality, set $G = 2^n$.
2. Evaluation – The $fitness$ of every individual of the population in each group is evaluated in this step.
3. Communication – Send the top best B individuals of the jth group to the qth groups to substitute B individuals in each receiving group randomly for every R generations, i.e., receive some information from the other groups but keep the same population size. Here, $q = j \oplus 2^i$, $j = 0, 1, ..., G-1$ and $i = 0, 1, ..., n-1$.
4. Reinitialization – After T generations, the convergence of each subpopulation is checked for every L generations. If top I individuals in the subpopulation are the same, then keep one best individual and this best individual is used as the template to initialize the subpopulation by swapping the genes W times in the chromosome for each new individual randomly. W is also generated randomly.
5. Selection – Set the number of survivors within each group to $P * P_{si}$ where P_{si} is the survival rate for the ith subpopulation, $i = 1, 2, ..., G$. For $r = 1$ to $P * P_{si}$, randomly choose M individuals from the group and select the best of these M individuals as a survivor. This selection scheme is also used in the Crossover step and Mutation step to select parents and candidates for crossover and mutation.

6. Crossover – The uniform order-based crossover technique [7] is used to produce the next generation from the selected parents for each group. $P * P_{ci}$ individuals for each group are generated in this step, where P_{ci} is the crossover rate for the ith subpopulation, $i = 1, 2, ..., G$. Several *gene* positions of the *chromosome* are chosen randomly and the order in which these *genes* appear in the second parent is imposed on the first parent to produce offspring. The *genes* in the other positions are the same as the first parent.

7. Mutation – The *genes* (or indices) in the *chromosomes* of the population are mutated according to the mutation rate P_{mi}, $i = 1, 2, ..., G$. Here, the total number of mutations for each group is set to group population size P * mutation rate P_{mi}. The mutation is only operated by exchanging two indices randomly in each group. Here, $P_{si} + P_{ci} + P_{mi} = 1$, $i = 1, 2, ..., G$.

8. Termination – Step 2 to step 7 are repeated until the predefined *fitness* or the number of generations have been reached. After termination, the optimal codevector indices are generated from the best individual for all groups.

4 Experimental Results

Experiments were carried out to test the performance of the parallel genetic algorithm, parallel genetic algorithm with the scheme of reinitialization, multi-mutation rates and multi-crossover rates, and average distortion of the random assignment for 32 codevectors. The performance is measured in terms of the average distortion using Eq. 8 compared with the average distortion of the random assignment for any bit error using Eq. 7. The average distortion instead of its inverse is used as the *fitness* in the parallel genetic algorithm to test the worst case of the random assignment. The distribution of the codevector probability is set to a uniform distribution. The parameter values used in the parallel genetic algorithm for the group population size P, the number of groups G, the predefined number of generations, the survival rate P_s, the crossover rate P_c, the mutation rate P_m, the number of individuals for selection M, the number of top best for communication B and the number of generations for communication R are 50, 8, 500, 0.5, 0.4, 0.1, 3, 1 and 50 respectively. For the parallel genetic algorithm with the scheme of reinitialization, multi-mutation rates and multi-crossover rates, the parameter values used for T, L, I, the maximum of W, the mutation rates P_{mi}, the crossover rates P_{ci} are 200, 10, 25, 3, $\{0.1, 0.2, 0.3, 0.4, 0.4, 0.3, 0.2, 0.1\}$ and $\{0.4, 0.3, 0.2, 0.1, 0.1, 0.2, 0.3, 0.4\}$. The parallel genetic algorithm is refered to as PGA and the parallel genetic algorithm with the scheme of reinitialization is called PGA-RI. The detail results of the experiments for 0.01 bit error probability are depicted in Table 3.

The experimental results of the parallel genetic algorithm in codebook index assignment for different population sizes are shown in Fig. 1. The average distortion decreases with increase in the population size. This result is reasonable because for more individuals, the parallel genetic algorithm will provide more possible solutions. Experimental results for the bit error probability from 0.01 to 0.3 for 32 codewords are depicted in Fig. 2. The spirit of the parallel genetic algorithm is not only to accelerate the speed of running time, but also to produce improved index assignments. In order to reach these objectives, the communication between groups should be operated for some fixed generations. By sending some top best individuals in the current group to the neighbouring groups, the problem of being trapped in the local optimum due to convergence in an earlier generation can be avoided because some promising individuals are migrated from the other groups to replace some worse individuals in the current group. Experiments have also been carried out to test performance in the separation of the groups. The number of possible solutions that the parallel genetic algorithm provides is $P \times G \times N_g$, where P, G and N_g are the group population size, the number of groups and the number of generations, respectively. The comparisons in the performance of the separating groups are based on the same total number of possible solutions, i.e., $P \times G \times N_g$ is kept constant. The total number of individuals of the population are separated into 8 groups, 4 groups, 2 groups and 1 group (standard genetic algorithm) and the group population sizes are 50, 100, 200 and 400, respectively. The other parameter values used for the predefined number of generations N_g, the bit error probability, the survival rate P_s, the crossover rate P_c, the mutation rate P_m, the number of individuals for selection M, the number of top best for communication B and the number of generations for communication R are 500, 0.01, 0.5, 0.4, 0.1, 3, 1 and 50 respectively. The experimental results for 32 codewords are shown in Fig. 3. The more groups are used, the better result is generated.

random	0.12900		
Seed	PGA-RI	PGA	Worst Case
1	0.057223	0.057117	0.216169
2	0.057094	0.057323	0.216480
3	0.057097	0.057218	0.216682
4	0.056864	0.057010	0.216950
5	0.057209	0.057371	0.217128
6	0.057125	0.057040	0.216832
7	0.056940	0.057020	0.217046
8	0.057149	0.058332	0.217064
9	0.057021	0.057118	0.216629
10	0.057372	0.057901	0.216559

Table 3: Ten runs of PGA-RI, PGA and the worst case

5 REFERENCES

1. Pan, J. S., McInnes, F. R., and Jack, M. A., "Application of Parallel Genetic Algorithm and Property of Multiple Global Optima to VQ Codevector Index Assignment", IEE Electronics Letters, Vol. 32, No. 4, 296–297, 1996

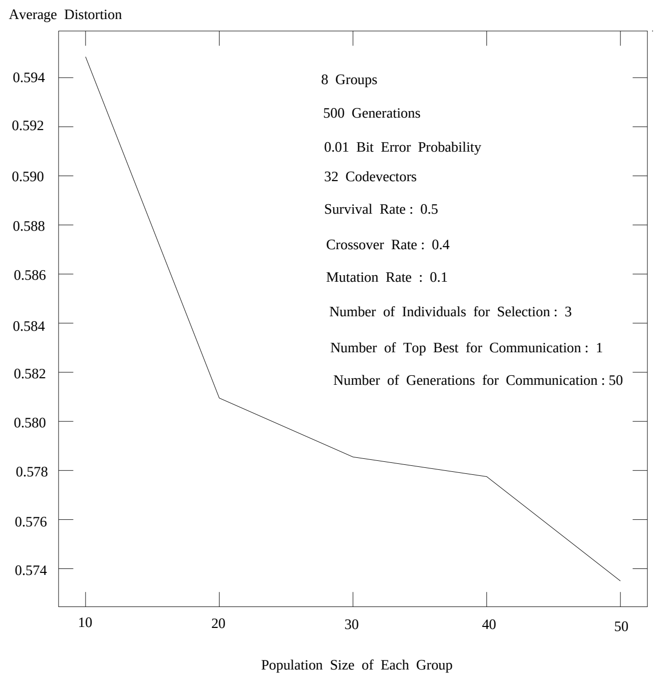

Figure 1: Average distortion of parallel genetic algorithm in codebook index assignment for different population size

2. Gray, R. M., "Vector Quantization", IEEE ASSP Magazine, 4–28, Apr. 1984

3. Zeger, K., and Gersho, A, "Pseudo-Gray Coding", IEEE Trans. on Communications, Vol. 38, No. 12, 2147–2158, Dec. 1990

4. Farvardin, N., "A Study of Vector Quantization for Noisy Channels", IEEE Trans. on Information Theory, July 1990, Vol. 36, No. 4, 799–809, July 1990

5. Wu, H. S., and Barba, J., "Index Allocation in Vector Quantization for Noisy Channels", IEE Electronics Letters, Vol. 29, No. 15, 1317–1319, 1993

6. Goldberg, D. E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company, 1989

7. Davis, L., "Handbook of Genetic Algorithms", Published by Van Nostrand Reinhold, 1991

8. Fang, H. L., "Genetic Algorithms in Timetabling and Scheduling", Ph.D. Thesis, Department of Artificial Intelligence, University of Edinburgh, 1994

9. Cohoon, J. P., Hegde, S. U., Martine W. N., and Richards, D., "Punctuated Equilibria: A Parallel Genetic Algorithm", Proceedings of the Second International Conference on Genetic Algorithms, 148–154, July 1987

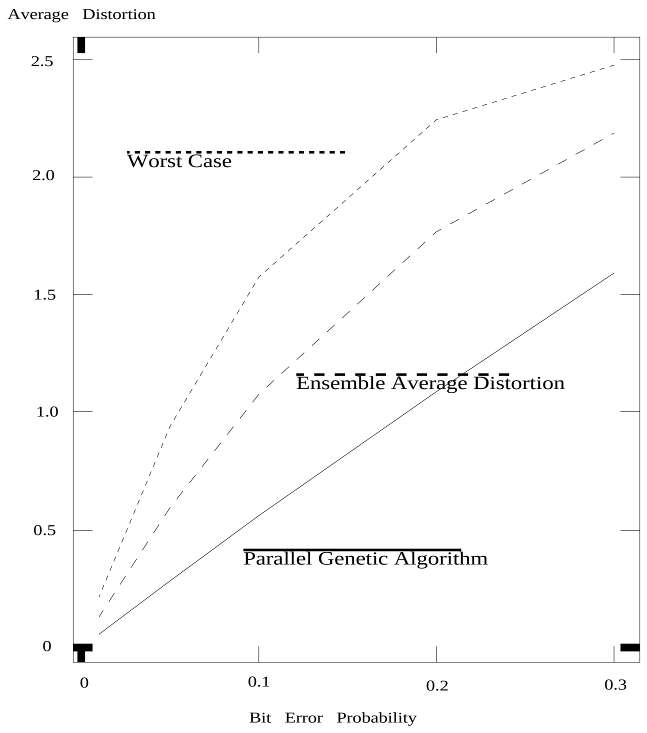

Figure 2: Average distortion of parallel genetic algorithm in codebook index assignment for different bit error probability

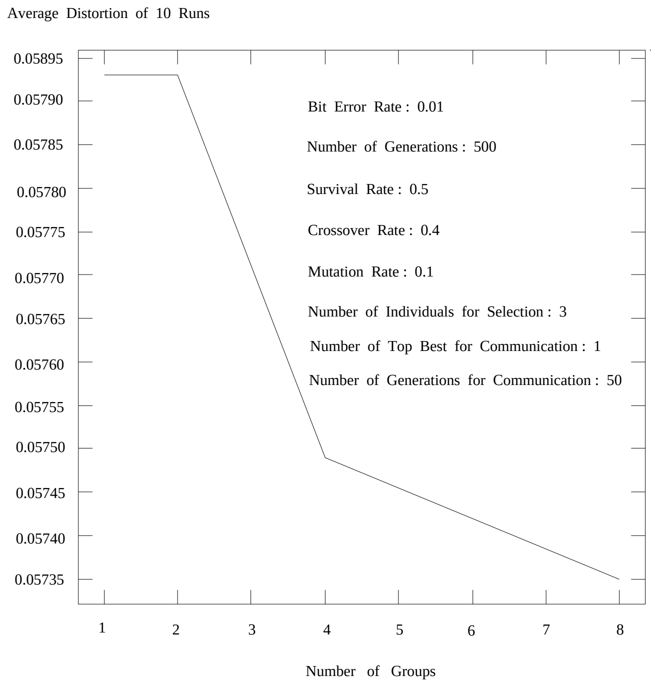

Figure 3: Average distortion of PGA in codebook index assignment for different numbers of groups