COMPARISON STUDY ON VQ CODEVECTOR INDEX ASSIGNMENT

J. S. Pan [*] , *C. S. Shieh* [*] *and S. C. Chu* [*, **]

[*] National Kaohsiung Institute of Technology, Taiwan
[**] University of South Australia, Australia

ABSTRACT

Vector quantization is a popular technique in low bit rate coding of speech signal. The transmission index of the codevector is highly sensitive to channel noise. The channel distortion can be reduced by organizing the codevector indices suitably. Several index assignment algorithms are studied comparatively. Among them, the index allocation algorithm proposed by Wu and Barba is the fastest method but the channel distortion is the worst one. The proposed parallel tabu search algorithm reach the best performance of channel distortion.

1.INTRODUCTION

Vector quantization (VQ) [1] is a widely used technique for data compression. The binary indices of the optimally chosen codevectors are sent to the destination. A vector $X = \{x^1, x^2, \cdots, x^k\}$ 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, $C = \{c_1, c_2, \cdots, c_N\}$, the partitioned set consisting of subspaces of the k-dimensional Euclidean space R^k, $S = \{s_1, s_2, \cdots, s_N\}$, and the mapping function Q(·):

$$Q(x) = C_i \ , \ if \qquad X \in S_i \tag{1}$$

The sets C and the partitioned set S_i satisfy

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

and

$$S_i \bigcap S_j = \phi \ \ if \qquad i \neq j \tag{3}$$

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

$$i = \arg\min_p \sum_{l=1}^{k} (x^l - C_p^l)^2 \tag{4}$$

only the index i is transmitted over the channel to the receiver. The channel noise will induce channel errors in the communication such that the index i is changed to index j. Thus, distortions are introduced in the decoding step. Distortion due to an imperfect channel can be reduced by assigning suitable indices to codevectors. The number of combination of indices to codevectors is N! since the number of codevectors is N. To test N! assignment is NP-hard problem.

The binary switching algorithm (BSA) [2] is proposed to improve the codevector index assignment by Zeger and Gersho. The main idea of binary switching algorithm is to calculate the expected distortion due to the single bit error in the index of codevectors for every index swapping and swap the index pair that makes the largest improvement in distortion. Obviously, the binary switching algorithm is the descent algorithm. It is difficult to reach the global optimum and always get trap in the local optimum.

The simulated annealing technique is applied to design the codevector indices by Farvardin [3]. An initial temperature is set and the initial state b of the indices for codevectors is chosen at random. Randomly choose annother state b' (perturbation of state b) and calculate $\delta D = D(b') - D(b)$. If $\delta D < 0$, replace b by b'; otherwise replace b by b' with probability $\exp(\frac{-\delta D}{T})$. If the number of average distortion drops exceeds a prescribed number or if too many unsucessful perturbations occur, then check the termination condition. The procedure will be terminated if the temperature is below some prescribed freezing temperature or a stable state is reached.

Wu and Barba [4] developed an efficient index allocation algorithm by using the information of a priori probability of codevectors which is referred to as IAP in this paper. Potter and Chiang [5] adopted a minimax design criterion instead of the mean squared error for codevector index assignment. The worst case performance is greatly improved which maintains good average performance by applying the minimax design criterion. Channel optimized vector quantization was also developed for indices assignment to get promising results [6,7]. Both the codebook generation and the codevector indices assignment are optimized together by iterative algorithm in the channel-optimized vector quantizer which is different from the separated optimization approaches.

Genetic algorithms [8,16-18] are adaptive methods which can

be used in the search and optimization problems. In genetic algorithms, a set of solutions to a problem is called chromosomes. A chromosome (string of solution) is composed of genes. Usually, the individual of the whole population contains only one chromosome. The performance of the solution is called fitness. The fitness of chromosomes are evaluated and ordered, then new chromosomes are produced by using the selected candidates as parents and applying mutation and crossover operations. The new set of chromosomes is then evaluated and ordered again. This cycle continues until a suitable solution is found. Data parallelism can be easily applied to genetic algorithms by dividing the population into several groups and running the same algorithm for each group at the same time using different processors which is called parallel genetic algorithm (PGA). The purpose of applying parallel processors to genetic algorithms is more than just a hardware accelerator. Rather a distributed formulation is developed which gives better solutions with less computation. In order to reach this function, the communication among these groups is executed for some fixed generations, i.e., the parallel genetic algorithm periodically selects promising individuals from each subpopulation and migrates them to different subpopulations. With the migration (communication), each subpopulation will receive some new and promising chromosomes to replace the worst chromosomes in this subpopulation. This helps to avoid premature convergence. The parallel genetic algorithm was applied to improve the codevector indices assignment by Pan et. al. [9,10]. The chromosome is composed of the string of the codevector indices. Several population groups are generated randomly. The individuals in each group are evaluated. Then the selection, crossover and mutation schemes are applied to each group separately. For some fixed generations, some promising individuals in each group are migrated to the neighbor groups. This procedure guarantees a better optimum will be reached easily from the experiments.

Assume that N codevectors C_i, i=1,2,…,N, are assigned codevector indices with an m bit string $b(c_i)$, where $N = 2^m$. Let $P(c_i)$ and $d(c_i, c_j)$ denote the probability of sending codevector C_i and the distortion between codevector C_i and C_j, i, j=1,2,…,N. A memoryless binary symmetric channel with bit error probability ε is simulated in this paper. For a 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 derived as [10]

$$D_c = \frac{1-(1-\varepsilon)^m}{N-1} \sum_{i=1}^{N} P(c_i) \sum_{j=1}^{N} d(c_i, c_j) \tag{5}$$

The objective performance for the transmission of indices $b(c_i)$, i=1,2,…,N, can be written as

$$D = \sum_{i=1}^{N} P(c_i) \sum_{i=1}^{m} \varepsilon^l (1-\varepsilon)^{m-1} \sum_{b(c_j) \in N^l(b(c_i))} d(c_i, c_j) \tag{6}$$

where $N^l(b(c_i)) = \{b(c_j) \in I, H(b(c_i), b(c_j)) = l\}$, is the lth neighbour set of $b(c_i)$. If the channel bit error probability ε is assumed to be sufficiently small ($m\varepsilon << 1$), then the error probability due to more than one bit error can be ignored and the bit error probability of the channel model can be expressed as

$$P(b(c_j) / b(c_i)) = \begin{cases} \varepsilon, & H(b(c_i), b(c_j)) = 1 \\ 1 - m\varepsilon, & H(b(c_i), b(c_j)) = 0 \\ 0, & H(b(c_i), b(c_j)) > 1 \end{cases} \tag{7}$$

Based on this channel model, the average distortion caused by the channel noise for a given assignment of indices, $b(b(c_1), b(c_2), ..., b(c_N))$, can be expressed as

$$D = \sum_{i=1}^{N} \sum_{j=1}^{N} P(c_i) P(b(c_j) / b(c_i)) d(c_i, c_j) \tag{8}$$

$$= \varepsilon \sum_{i=1}^{N} P(c_i) \sum_{j: H(b(c_i), b(c_j))=1} d(c_i, c_j), \tag{9}$$

and the ensemble average distortion is

$$\overline{D} = \frac{\varepsilon}{N!} \sum_{i=1}^{N} P(c_i) \sum_{all\ b} \sum_{j: H(b(c_i), b(c_j))=1} d(c_i, c_j) \tag{10}$$

$$= \frac{\varepsilon m}{N-1} \sum_{i=1}^{N} P(c_i) \sum_{j=1}^{N} d(c_i, c_j). \tag{11}$$

In this paper, the tabu search approach [11-14] is paralleled and applied to the codevector index assignment. Binary switching algorithm (BSA), simulated annealing (SA), parallel genetic algorithm (PGA), IAP and parallel tabu search (PTAB) approach are comparatively studied in this paper. The main contribution of this paper is to distribute the work of cdevector index assignment to several processors by using the tabu search approach so as not only to accelerate the computation time but also to reduce the channel distortion.

2.TABU SEARCH APPROACH

The tabu search approach was proposed by Glover [11]. The basic idea of the tabu search is to explore the search space of all feasible solutions by a sequence of moves. The spirit of this method is embedded in its short-term memory process. The elements of the move from the current solution to its selected neighbor are partially or completely recorded in the tabu list for the purpose of forbidding the reversal of the replacement in a number of future iterations. The search will cycle between the first encountered local minimum and its neighbor without this assurance.

The tabu search scheme starts with test solutions generated randomly and evaluated the objective function for these solutions. If the best of these solutions is not tabu or if it is tabu, but satisfies the aspiration criterion, then select this solution to be the new current solution to generate test solutions

for next iteration. The process is terminated if the predefined objective value or the number of iterations have been reached. It is called aspiration criterion if the test solution is a tabu solution but the objective value is better than the best value of all iterations.

3.PAPRLLEL TABU SEARCH ALGORITHM

Let $S_{t,u}$, $s_{c,u}$ and $s_{b,u}$ be the test, current and best solutions and $V_{t,u}$, $v_{c,u}$ and $v_{b,u}$ denote the corresponding test values, best values of current iteration and best value of all iterations for the *uth* group, respectively. Set the number of groups is *G* and $G = 2^n$. $S_{t,u} = \{s_{t,u}^1, s_{t,u}^2, ..., s_{t,u}^{N_s}\}$, $s_{t,u}^i = \{s_{t,u}^i(1), s_{t,u}^i(2), ..., s_{t,u}^i(N)\}$, $1 \le i \le N_S$, $V_{t,u} = \{v_{t,u}^1, v_{t,u}^2, \ldots, v_{t,u}^{N_s}\}$, $s_{c,u} = \{s_{c,u}(1), s_{c,u}(2), ..., s_{c,u}(N)\}$, and $s_{b,u} = \{s_{b,u}(1), s_{b,u}(2), ..., s_{b,u}(N)\}$, for the *uth* group, where N_s is the number of test solutions and *N* is the number of codevector indices. The initial test solutions are generated randomly. After the first iteration, the test solutions are generated from the best solution of current iteration as shown in step 2 of the following algorithm. The tabu list memory only store the swapping indices. It is a tabu condition if the swapped indices to generate the test solution from the best solution of current iteration is the same as any record in the tabu list memory. The size of the tabu list memory is $2 \times N_s$. The aspiration level is also applied in this algorithm. The algorithm is described as follows:

Step 1. Set the tabu list size T_s, number of test solutions N_s and the maximum number of iterations I_m. Set the iteration counter *i*=1 and the tabu list length $t_l = 1$. Generate N_s initial solutions $S_{t,u} = \{s_{t,u}^1, s_{t,u}^2, ..., s_{t,u}^{N_s}\}$ for each group randomly, calculate the corresponding objective values $V_{t,u} = \{v_{t,u}^1, v_{t,u}^2, ..., v_{t,u}^{N_s}\}$ using Eq. 6 (for multiple bit errors) or Eq. 9 (for single bit error) and select the current best solution $s_{c,u} = s_{t,u}^{j_u}$, $j_u = \arg\min_l v_{t,u}^l$ for the *nth* group, $1 \le l \le N_s$. Set $s_{b,u} = s_{c,u}$ and $v_{b,u} = v_{c,u}$.

Step 2. Given the current best assignment $s_{c,u}$, generate two random integers r_1 and r_2 for each test solution, $1 \le r_1 \le N$, $1 \le r_2 \le N$, $r_1 \ne r_2$, *N* is the number of codevector indices. Generate the test solutions by swapping $s_{c,u}(r_1)$ and $s_{c,u}(r_2)$. Calculate the corresponding objective values $v_{t,u}^1, v_{t,u}^2, ..., v_{t,u}^{N_s}$.

Step 3. Sort $v_{t,u}^1, v_{t,u}^2, ..., v_{t,u}^{N_s}$ in an increasing order. From the best test solution to the worst test solution, if the test solution is a non-tabu solution or it is a tabu solution but the objective value is better than the best value of all iterations (aspiration level), then choose this test solution as the current best solution and go to step 4; otherwise, try the next test solution. If all test solutions are tabu solutions, then go to step 2.

Step 4. Set the best test solution of the *jth* group to the *qth* groups to substitute the one test solution in each receiving group randomly for every *R* iterations. Here, $q = j \oplus 2^i$, *j*=0,1,…,*G*-1 and *i*=0,1,…,*m*-1. *m* is the bit number for the index.

Step 5. Set $s_{b,u} = s_{c,u}$ and $v_{b,u} = v_{c,u}$. Insert the swapped indices of the current best solution to the tabu list. Set the inserting point of the tabu list $t_l = t_l + 1$. If $t_l > T_s$, set $t_l = 1$. If $i < I_m$, set *i*=*i*+1 and go to step 2; otherwise, record the best codevector index assignment and terminate the program.

4.EXPERIMENTS AND CONCLUSIONS

The test material for these experiments is Gauss-Markov source which is of the form:

$$y_n = \alpha y_{n-1} + w_n \tag{12}$$

where w_n is a zero-mean, unit variance, Gaussian white noise process, with $\alpha = 0.5$. Applying LBG algorithm [15], 7500 vectors with 8 dimensions are used to generate 32 and 64 codevectors. The single bit error for each codevector index is assumed. Eq. 9 is used for the evaluation of the channel distortion. The ensemble average distortion can be computed by using Eq 11.

Experiments were carried out to test the performance of the binary switching algorithm (BSA), simulated annealing (SA), parallel genetic algorithm (PGA), IAP algorithm and the parallel tabu search algorithm (PTABU) for using in codevector index assignment for 32 and 64 codevectors. The parameter values setting for SA and PGA are the same as in papers [3] and [9]. For the parallel tabu search approach, the parameter values used for the tabu list size T_s, the number of groups G, the number of test solutions N_s, the number of iterations for communication *R* and the maximum number of iterations I_m are 20, 4, 100, 20 and 500, respectively. Table 1 and Table 2 show the mean squared error of the PTABU, PGA, BSA. SA, IAP and the ensemble average distortion with 0.01 bit error probability for 10 runs for 32 and 64 codevectors.

For the limited experiments, the parallel tabu search algorithm may reach the best results. The performance of the parallel genetic algorithm is better than the binary switching algorithm for the 32 codevector indices assignment, but it is opposite for

64 codevector indices. The parallel genetic algorithm can be better than binary switching algorithm for more than 32 codevector by increasing the number of generations. Especially, both the PGA and PTABU can distribute the computation to several processors so as to speed the computation result. The IAP algorithm is the fastest method. The operation time for the IAP algorithm is no more than 1 sec, but the performance of the channel distortion is the worst one. For considering the running speed and the performance of the channel distortion, the better performance can be obtained by using the IAP algorithm firstly, then applying the PTABU algorithm.

Random	0.701206				
SEED	PTABU	PGA	BSA	SA	IAP
1	0.341860	0.373131	0.390412	0.361024	0.545358
2	0.347710	0.378316	0.375556	0.364418	
3	0.342562	0.392741	0.380347	0.363039	
4	0.346233	0.357870	0.391600	0.352267	
5	0.342963	0.371268	0.388615	0.344427	
6	0.340755	0.369513	0.363033	0.347041	
7	0.341422	0.368067	0.363809	0.373717	
8	0.344369	0.379738	0.399675	0.360959	
9	0.341467	0.366208	0.412873	0.348636	
10	0.344491	0.376029	0.364416	0.352290	
Average	0.343383	0.373288	0.383034	0.356782	0.545358

Table 1: Performance comparison of PTABU, PGA, BSA, SA, IAP and random algorithms for 32 codevectors.

Random	0.903263				
SEED	PTABU	PGA	BSA	SA	IAP
1	0.427747	0.537973	0.454295	0.458354	0.655748
2	0.432621	0.523357	0.469178	0.459786	
3	0.435934	0.530668	0.447620	0.461031	
4	0.430543	0.519685	0.435605	0.477968	
5	0.437133	0.527924	0.457320	0.440630	
6	0.434815	0.532623	0.446801	0.459827	
7	0.432443	0.533488	0.439722	0.467991	
8	0.429790	0.536568	0.440468	0.455524	
9	0.427665	0.536530	0.452813	0.443455	
10	0.429963	0.532848	0.466651	0.459152	
Average	0.431865	0.531168	0.451047	0.458372	0.655748

Table 2: Performance comparison of PTABU, PGA, BSA, SA, IAP and random algorithms for 64 codevectors.

5.ACKNOWLEDGEMENT

The authors would like to acknowledge the financial support for this work by the National Science Council of R.O.C.. This material is part of the work under the Grant No. 87-2213-E-151-007.

6.REFERENCES

1. Gray, R. M. "Vector Quantization," *IEEE ASSP Magazine*, 4-28, 1984.
2. Zeger, K. and Gersho, A., "Pseudo-Gray Coding," *IEEE Trans. on Communications*, Vol. 38, No. 12, 2147-2158, 1990.
3. Farvardin, N., "A Study of Vector Quantization for Noisy Channels," *IEEE Trans. on Information Theory*, Vol. 36, No. 4, 799-809, 1990.
4. Wu, H. S. and Barba, J., "Index Allocation in Vector Quantization for Noisy Channels," *IEE Electronics Letters*, Vol. 29, No. 15, 1317-1319, 1993.
5. Potter, L. C. and Chiang D. M., "Minimax Nonredundant Channel Coding," *IEEE Trans. on Communications*, Vol. 43, No. (2/3/4), 804-811, 1995.
6. Farvardin, N. and Vaishampayan, V., "On the Performance and Complexity of Channel-Optimized Vector Quantizers," *IEEE Trans. on Information Theory*, Vol. 37, No. 1, 155-160, 1990.
7. Kumazawa, H., Kasahara, M. and Namekawa, T., "A Construction of Vector Quantizers for Noisy Channels," *Electronics and Engineering in Japan*, 39-47, 1984.
8. Goldberg, D. E., *Genetic Algorithm in Search, Optimization and Machine Learning* , Addison-Wesley Publishing Company, 1989.
9. 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.
10. Pan, J. S., McInnes, F. R. and Jack, M. A., "VQ Codevector Index Assignment Using Genetic Algorithms for Noisy Channels," *Proc, of The Fourth International Conference on Spoken Language Processing*, 295-298, 1996.
11. Glover, F., "Tabu Search, Part Ⅰ," *ORSA Journal on Computing*, Vol. 1, No. 3, 190-206, 1989.
12. Glover, F., "Tabu Search, Part Ⅱ," *ORSA Journal on Computing*, Vol. 2, No. 1, 4-31, 1990.
13. Skorin-Kapov, J., "Tabu Search Applied to The Quadratic Assignment Problem", *ORSA J. Computing*, Vol. 2, No. 1, 33-45, 1990.
14. Glover, F., "Artificial Intelligence, Heuristic Frameworks and Tabu Search," *Managerial And Decision Economics*, Vol. 11, 365-375, 1990.
15. Linde, Y., Buzo, A. and Gray, R. M., "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Communication*, Vol. 28, No. 1, 84-95, 1980.
16. Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.
17. 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, 1987.
18. Pan, J. S., McInnes, F. R. and Jack, M. A., "VQ Codebook Design Using Genetic Algorithms," *IEE Electronics Letters*, Vol. 31, No. 17, 1418-1419, 1995.