Comparison of Nonuniform Optimal Quantizer Designs for Speech Coding With Adaptive Critics and Particle Swarm

Ganesh Kumar Venayagamoorthy, *Senior Member, IEEE*, and Wenwei Zha, *Student Member, IEEE*

***Abstract*—This paper presents the design of a companding nonuniform optimal scalar quantizer for speech coding. The quantizer is designed using two neural networks to perform the nonlinear transformation. These neural networks are used in the front and back ends of a uniform quantizer. Two approaches are presented in this paper namely adaptive critic designs and particle swarm optimization, aiming to maximize the signal-to-noise ratio. The comparison of these optimal quantizer designs over a bit-rate range of 3–6 is presented. The perceptual quality of the coding is evaluated by the International Telecommunication Union's Perceptual Evaluation of Speech Quality standard.**

***Index Terms*—Adaptive critic designs (ACDs), neural networks, particle swarm optimization (PSO), perceptual evaluation of speech quality (PESQ), quantization, speech coding.**

I. Introduction

QUANTIZATION is the representation of a large set of information elements with a much smaller set. It is a crucial link in speech coding. After the original sound signal sequence is quantized, some information is lost. The key task of quantization design is to minimize the information loss. Two categories of criterions are applied to evaluate the information loss, namely objective criterion and subjective criterion. Objective criterion is a quantization distortion such as signal-to-noise ratio (SNR) or sum squared error. Subjective criterion is how the human perceives about the quantized result, i.e., how the quantized signal sounds to the human ears. Generally, a nonuniform quantization causes less information loss than uniform quantization, especially for small quantization resolutions or bit rates.

This paper presents two new strategies for optimal nonuniform scalar quantizer designs for small bit rates (three to six). The quantizer is designed to be optimal in SNR rather than in sum squared error, because the SNR is more relevant to perceptual quality. The ultimate evaluation of quantization quality is to how the quantized speech signal resembles the original speech signal in a perceptual way [1]. Normally, subjective experiments like Mean Opinion Score (MOS) are used to evaluate the subjective quality, but they are very expensive to implement. A simpler method is to use subjective models like International Telecommunication Union's (ITU) Perceptual Evaluation of Speech Quality (PESQ) standard to evaluate the quantization quality.

The rest of this paper is organized as follows. The nonuniform scalar quantizer design using neural networks is described in Section II. Implementing the quantizer using adaptive critic designs (ACDs) is described in Section III. Section IV describes the particle-swarm-optimization (PSO)-based approach to the quantizer design. Section V briefly introduces the ITU PESQ standard. Simulation results, comparison, and discussions are given in Section VI. Finally, the conclusions are given in Section VII.

II. Nonuniform Scalar Quantizer Using Neural Networks

A scalar quantizer of size N is a mapping from a real number $x \in \boldsymbol{R}$ into a finite set Y, i.e., codebook containing N output values (also known as reproduction points or codewords) y_i [2]. The quantization can be denoted as $Q(\bullet)$

$$y(n) = Q\left(x(n)\right) \tag{1}$$

with its bit rate or resolution defined as

$$R = \log_2 N. \tag{2}$$

The SNR is defined as

$$\text{SNR} = 10 \cdot \log_{10} \frac{\text{var}\left[x(n)\right]}{\text{var}\left[x(n) - y(n)\right]} \tag{3}$$

where $x(n)$ is the original speech signal sequence, $y(n)$ is the output of the nonuniform quantizer, and var$(\bullet)$ stands for variance.

In general, a uniform quantization is not the most effective way to achieve a good performance. For a given number of quantizing intervals, taking into account the input probability density, nonuniform spacing of the decision levels can yield lower quantizing noise and less sensitivity to variations in input signal statistics. There are two approaches to design optimal

Paper MSDAD-06-19, presented at the 2005 Industry Applications Society Annual Meeting Hong Kong, October 2-6, and approved for publication in the IEEE Transactions on Industry Applications by the Industrial Automation and Control Committee of the IEEE Industry Applications Society. Manuscript submitted for review October 15, 2005 and released for publication August 9, 2006. This work was supported by the National Science Foundation (NSF) under CAREER Grant ECS 0348221 and Grant ECS 0529292.

G. K. Venayagamoorthy is with the Real-Time Power and Intelligent Systems Laboratory, Department of Electrical and Computer Engineering, University of Missouri, Rolla, MO 65409-0249 USA (e-mail: gkumar@ieee.org).

W. Zha was with the Real-Time Power and Intelligent Systems Laboratory, Department of Electrical and Computer Engineering, University of Missouri, Rolla, MO 65409-0249 USA. He is now with ESS Tech Inc., Fremont, CA 94538 USA (e-mail: Wenwei.zha@esstech.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIA.2006.885897

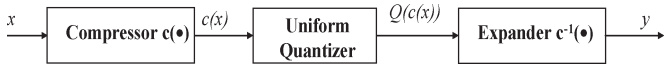


Fig. 1. Basic structure of nonuniform quantizer.

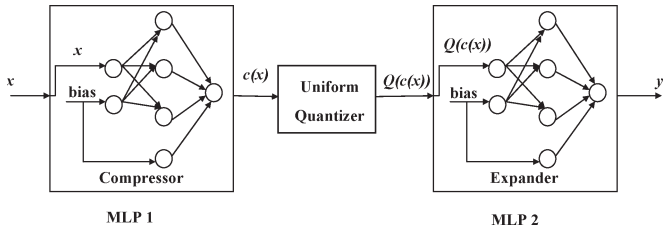


Fig. 2. MLP-based nonuniform quantizer.

nonuniform quantizer, i.e., to find the optimum quantization codebook. One is an iterative procedure for the exact solution based on the amplitude probability density function of the signal. Lloyd algorithm is based on this approach and is widely used. However, it is computationally intensive. The other approach is to compress the input signal x with a nonlinear transformation $c(\bullet)$, quantize the compressed signal $c(x)$ using a uniform quantizer, and expand the quantized signal $Q(c(x))$ with another nonlinear transformation, that is the inverse of $c(\bullet)$ [2]. The block diagram of this companding (compressing and expanding) nonuniform quantizer is shown in Fig. 1.

In practice, a widely used nonuniform quantizer, the logarithm quantizer, has the structure of Fig. 1. The compressor is logarithm transformation, and the expander is exponential transformation. The North American PCM standard μ-Law logarithm quantization ($\mu = 255$) is given by (4) and (5)

$$c(x) = A\frac{\ln(1+255|x|/A)}{\ln(1+255)}\text{sgn}(x), \qquad |x| \le A \tag{4}$$

$$y = \frac{A}{255}\left[\exp\left(\frac{\ln(1+255)\cdot \text{abs}\,(c(x))}{A}\right) - 1\right] \times \text{sgn}\,(c(x)), \qquad c(x) \le A. \tag{5}$$

For the first approach, a complete new iterative procedure needs to be run every time the quantization resolution changes. For the logarithm quantizer, it works well only with a large quantization resolution. By using neural networks to perform the nonlinear transformations, it is possible to design an optimal nonuniform quantizer which works well for a given bit-rate range and causes less information loss than the logarithm quantizer.

Multilayer perceptrons (MLPs) feedforward neural networks are known to be universal approximators [3], and are used in this paper to carry out the nonlinear transformations. The structure of the MLP-based companding quantizer is illustrated in Fig. 2.

Compared with Fig. 1, the compressor and the expander are replaced by MLP 1 and MLP 2, respectively. The MLPs are each of size $1 \times 3 \times 1$ with a bias in the input and hidden layer, thus a total of ten weights per MLP. The activation function of the hidden layer neurons is sigmoid, defined by the following equation:

$$f(x) = \frac{1}{1+e^{-x}}. \tag{6}$$

III. Implementation With ACD Approach

ACDs are neural-network-based optimization techniques, combining concepts of reinforcement learning and approximate dynamic programming [4]. In this paper, the heuristic dynamic programming (HDP) [5] approach of the ACD is used without a model, also known as the action-dependent HDP (ADHDP). The ADHDP-based nonuniform quantizer design is illustrated in Fig. 3.

There are three neural networks: one action network (MLP1 in Fig. 2) which performs the nonlinear transformation on the original signal $x(t)$ and outputs $A(t)$, one inverse action network (MLP2 in Fig. 2) which performs the inverse nonlinear transformation on the quantized signal $Q(t)$ and produces the final degraded signal $y(t)$, and a critic network which approximates the cost-to-go function $J(t)$, the Bellman's equation of dynamic programming is given as

$$J(t) = \sum_{k=0}^{\infty} \gamma^k U(t+k) \tag{7}$$

where $U(t)$ is the utility function and γ is a discount factor in (0, 1). The goal of ACDs is to minimize the cost-to-go function $J(t)$ over time with an optimal action network.

The signal $x(t)$ is divided into windows of N samples ($N = 50$ in this paper). The input of critic networks $A'(t)$ is defined as

$$A'(t) = \sum_{t}^{t+N-1} A(t). \tag{8}$$

The reason for dividing $x(t)$ into windows is that the SNR has statistical characteristic. It is meaningless to compute the SNR of the actual signal or the quantized signal over one or a few samples. This statistical characteristic requires that the SNR be calculated over a window with sufficient samples.

The quantization error is defined using the SNR of each window for different bit rate as follows:

$$\text{error}(t) = \frac{1}{4}\sum_{\text{bit}=3}^{6} 6.4 - \frac{\text{SNR(bit)} + 7}{\text{bit}} \tag{9}$$

where k denotes the number of window and bit is the quantization bit rate. SNR plus seven is approximately in direct ratio to bit rate. By dividing with bit, each bit rate has almost the same impact on the quantization error. This impact in quantity is around six, and in (9), it is subtracted by a constant 6.4 so that error(t) is a monotonically decreasing function of SNR. 6.4 is just an empirical value which works fine here. The utility function for the optimal quantizer design in (7) is given as

$$\begin{cases} U(1) = \frac{3}{2}\text{error}(1) & t = 1 \\ U(2) = \text{error}(2) + \frac{1}{2}\text{error}(1) & t = 2 \\ U(t) = \text{error}(t) + \frac{1}{4}\text{error}(t-1) + \frac{1}{4}\text{error}(t-2) & t \ge 3. \end{cases} \tag{10}$$

Two critic networks as shown in Fig. 3 are used to approximate $J(t-1)$ and $J(t)$, respectively. Both networks have the same weights but different inputs in time. The critic

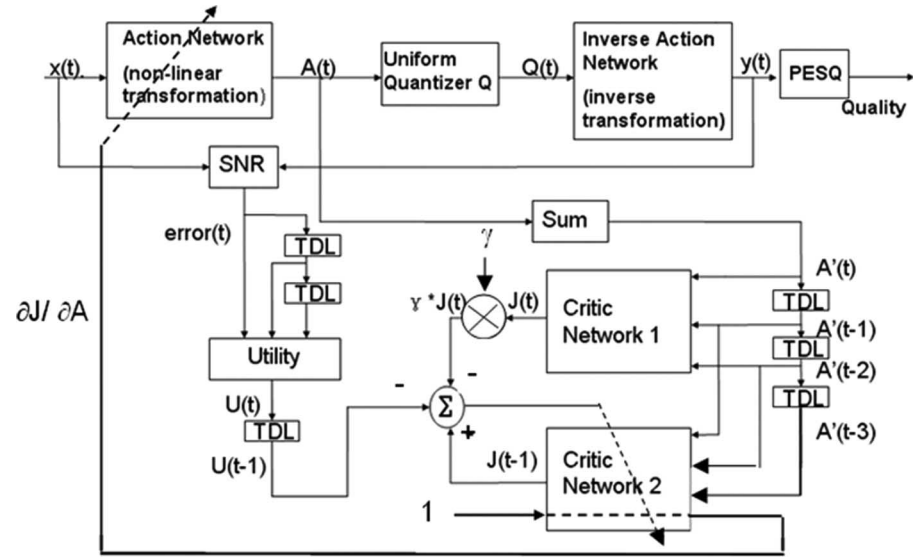


Fig. 3. Nonuniform quantizer implementation based on ACD technique.

network is trained to minimize the following error measured over time:

$$\|E_1\| = \sum_t E_1^2(t) \tag{11}$$

$$E_1(t) = J(t-1) - \gamma J(t) - U(t-1). \tag{12}$$

The weights of critic networks are updated online using the backpropagation algorithm. The change in weights is given as

$$\Delta W_c = -\eta_c E_1(t) \frac{\partial J(t)}{\partial W_c} \tag{13}$$

where η_c is the learning rate in the range of (0, 1).

The action network is trained to minimize $J(t)$. Using backpropagation, the weights of action network are updated as follows:

$$\begin{aligned} \Delta W_A &= -\eta_A \frac{\partial J(t)}{\partial W_A} \\ &= -\eta_A \frac{\partial J(t)}{\partial A(t)} \frac{\partial A(t)}{\partial W_A} \end{aligned} \tag{14}$$

so that the utility function is minimized, and thus, the SNR for each bit rate is maximized.

The inverse action network should perform exactly the inverse function of the action network. It is trained using backpropagation as illustrated in Fig. 4. The output of the inverse action network $x'(t)$ (shown as $y(t)$ in Fig. 3) should be trained to be identical or close as possible to the input of the action network $x(t)$.

IV. Implementation With PSO

PSO is a population-based evolutionary optimization technique developed by J. Kennedy and R. Eberhart in 1995, inspired by the social behavior of bird flocking or fish schooling [6]. PSO shares a lot of similarities with other evolutionary computation techniques like genetic algorithms. PSO does not involve crossover or mutation operators; rather, it has a memory, and tracks the best solution achieved in the past. PSO is attractive because it has few parameters to adjust, and it converges to better solutions faster and is less computationally intensive compared to many other methods. During the past few years, the PSO has been shown successful for many applications [7]. Several papers discuss how to apply the PSO in training neural networks and their advantages [8].

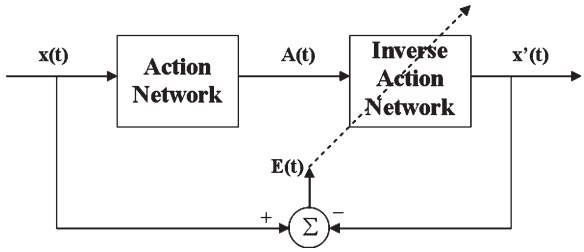


Fig. 4. Inverse action network training.

Like most evolutionary computation techniques, the PSO starts with a population of solutions, usually called particles, randomly initialized in the solution space. The particles search for the optima determined by a fitness function. Each particle representing one potential solution flies in the search space with a velocity adjusted according to the best position in its own flying experience (pbest) and the best position in all its companions' flying experience (gbest) or the best position in its neighbors' flying experience (lbest).

There are two versions of PSO—the global (gbest) and the local (lbest). The gbest version PSO is applied in this paper, and the procedure is as follows.

Step 1) Initially assign a population of particles with random positions (potential solutions) and velocities in d dimensions in the problem space. The initial pbest for each particle is set as its original position. Calculate the fitness of each particle and store the value. Find the best fitness among all particles and store the value and its corresponding position as the initial gbest.

Step 2) Update each particle's velocity V and position X according to

$$V(k+1) = w^* V(k) + c_1^* \text{rand}()^* (\text{pbest} - X(k)) + c_2^* \text{rand}()^* (\text{gbest} - X(k)) \quad (15)$$

$$X(k+1) = X(k) + V(k+1) \quad (16)$$

where w is the inertia weight, c_1 and c_2 are cognitive and social acceleration constants. In order to restrict the particles from traveling out of the solution space, a limit $V_{\max}$ is usually placed on the velocity. When the velocity exceeds this limit in any dimension, the value is set as the limit.

Step 3) Update pbest and gbest based on each particle's new position. Compare each particle's fitness evaluation with its pbest's fitness. If current fitness is better than pbest's, then set pbest to be the particle's current location and store the fitness value. Find the best fitness evaluation from each particle's pbest. If the value is better than gbest's fitness, then store this value and set gbest to be the location of pbest corresponding to this value.

Step 4) Repeat steps 2) and 3) until a criterion is met. The criterion is usually the maximum number of iterations, acceptable fitness of the gbest, or tolerable convergence of all particles.

In this paper, a population size of 25 is used. Each particle is a 20-dimensional vector representing the weights of the two MLPs. The fitness function is determined by the quantization SNR at different bit rates. The fitness function is given by the following equation:

$$\text{fitness} = 2000 - \sum_{\text{bit}=3}^{6} \left[\frac{\text{SNR(bit)}}{\text{bit}} \right]^3. \quad (17)$$

Dividing by bit, the SNRs for different bit rates are approximately normalized so that they have roughly the same impact on the fitness function. The cubic operation makes the fitness function nonlinear so it decreases fast with increasing SNRs. The value 2000 is just a bias value. The fitness function can be formulated differently. For example, the fitness function using square instead of cubic and using a different bias should also work for the problem studied here since it is also a nonlinear function monotonically decreasing on SNR. The flexibility of designing fitness function is another merit of the PSO.

Inertia weight w in (15) improves the performance of PSO algorithm [6] and in many applications is decreased linearly from about 0.9 to 0.4 during a PSO search. To insure the convergence of the PSO, it might be necessary to use a constriction factor [9]. Thus, (15) is changed to

$$V(k+1) = K\,[V(k) + c_3^* \text{rand}()^* (\text{pbest} - X(k)) + c_4^* \text{rand}()^* (\text{gbest} - X(k))] \quad (18)$$

where

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$$

and

$$\varphi = c_3 + c_4, \qquad \varphi > 4. \quad (19)$$

It is interesting to point out that although initial PSO applications often set $V_{\max}$ at about 10%–20% of the dynamic range on each dimension, the limit on $V_{\max}$ tends to be conceptually unnecessary by introducing the constriction factor. It is suggested in [10] to limit $V_{\max}$ to the dynamic range of dimension as a rule of thumb. Anyway, there is no limit on the dynamic range of the weights of the MLP neural networks in this paper.

The parameters used in the PSO algorithm in this paper are set as follows: 1) φ increases from 4.05 to 4.25 in (19), so that K decreases from 0.8 to 0.6; 2) $c_4 = c_3 = 0.5\varphi$ in (8); and 3) $w = K$, $c_1 = Kc_3$, and $c_2 = Kc_4$ in (16). A $V_{\max}$ of 5 works well in this paper. The reasons for choosing the above parameters are: 1) A linear decreasing inertia weight is used to improve the PSO performance. Although in [10] it is suggested that w be decreased from 0.9 to 0.4, the idea is to use a bigger w around 1.0 initially and use a smaller w around 0.5 in the end. The initial value of 0.8 and the final value of 0.6 work fine in this paper. 2) The constriction factor is used to ensure convergence. Normally, cognitive and social acceleration constants c_1 and c_2 are equal, so in (18) and (19), φ increases from 4.05 to 4.25 so that K as well as w decreases from 0.8 to 0.6.

Conceptually, the second MLP in Fig. 2 should perform exactly the inverse transformation of the first so its weights are fixed when the first MLP's weights are fixed. However, in this paper, all the 20 weights (MLP 1 and MLP 2) are to be searched for with PSO simultaneously. For one thing, it is easier to implement. Moreover, it might produce better result if the second MLP performs a slightly different function from the exact inverse function. This can be viewed as an improvement over the original definition of companding nonuniform quantizer: The second nonlinear transformation does not have to be exactly the inverse of the first transformation in order to minimize the quantization error (however, the result shows that the second MLP actually performs the inverse transformation).

V. PESQ

PESQ [11] is documented as ITU-T Recommendation P.862, which was prepared by ITU-T Study Group 12 (2001–2004) and approved under the WTSA Resolution 1 procedure on February 23, 2001. This Recommendation describes an objective method for predicting the subjective quality of speech coding. PESQ compares an original signal $X(t)$ with a degraded signal $Y(t)$, which is the quantizer output in this paper; and the output is a prediction of the perceived quality that would be given to $Y(t)$ by subjects in a subjective listening test, which uses a MOS-like scale ranging from 1(very bad) to 5(excellent).

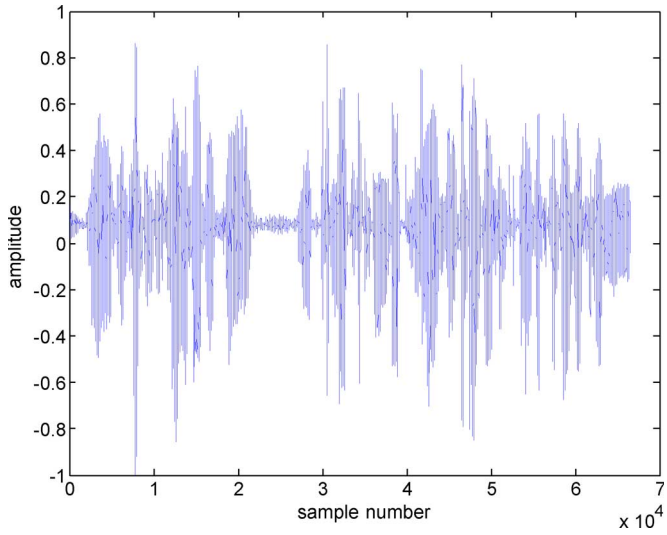


Fig. 5. Speech signal #1 amplitude plot.

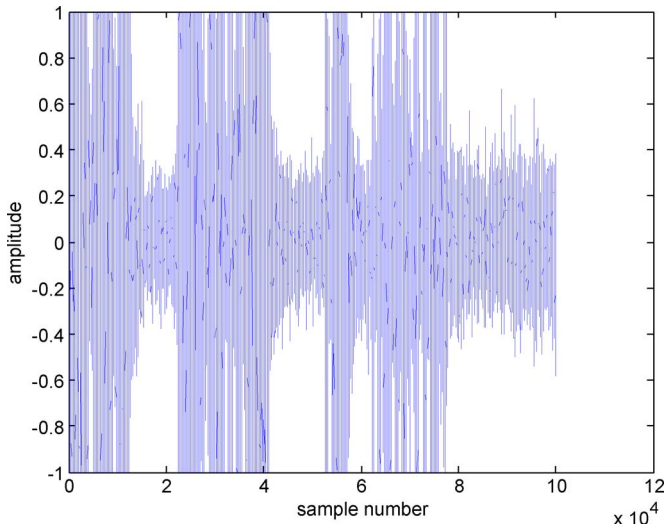


Fig. 7. Speech signal #3 amplitude plot.

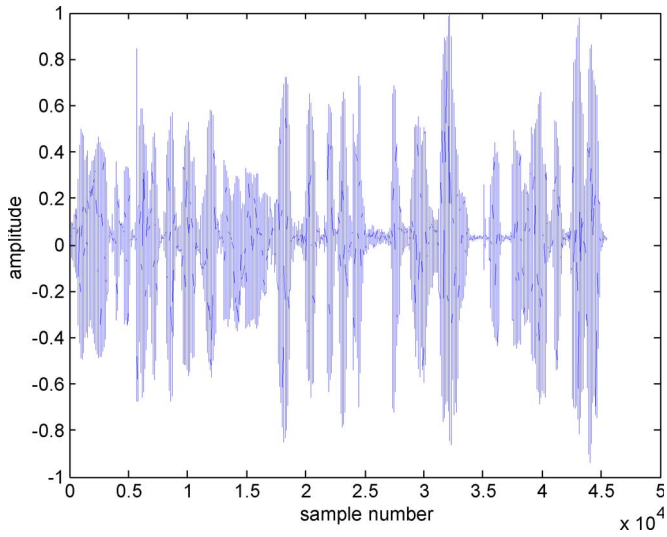


Fig. 6. Speech signal #2 amplitude plot.

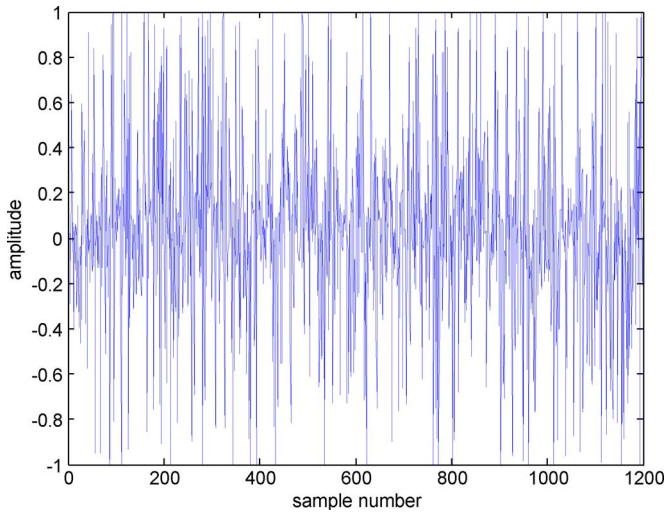


Fig. 8. Training speech signal amplitude plot.

VI. Results, Comparison, and Discussions

Three different short durations of speech of different people sampled at 8 KHz shown in Figs. 5–7 are used for the study. 1200 samples extracted randomly from the three speech signals are used as training data for the quantizer design. The amplitudes plot for the training data is shown in Fig. 8. The utility function with the ACD approach given in (10) and fitness function with the PSO approach given in (17) over iterations of training are shown in Figs. 9 and 10, respectively. Fig. 10 shows the fitness of the gbest particle.

In Figs. 9 and 10, it is observed that the fitness function in PSO and the utility function in ADHDP are decreasing over the number of iterations/epochs. Note that the fitness function and the utility function are based on different equations, so it does not make too much sense to compare the two Y-coordinates in the figures. Also, the Y-coordinates do not represent the SNR directly from (10) and (17), but this implies that the SNR is actually increasing.

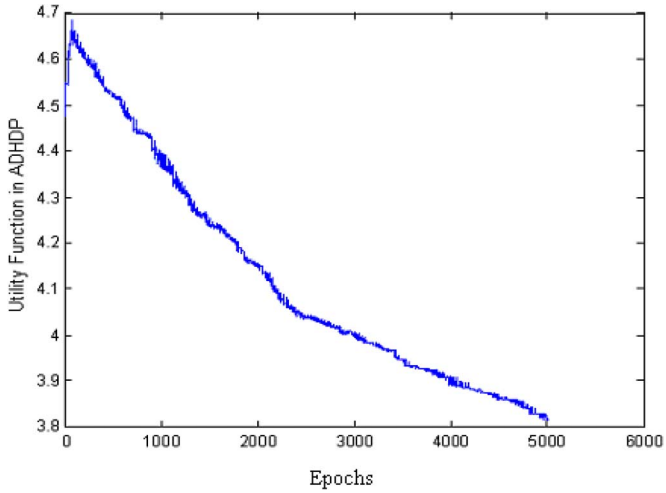


Fig. 9. Utility function with the ACD approach over iterations of training.

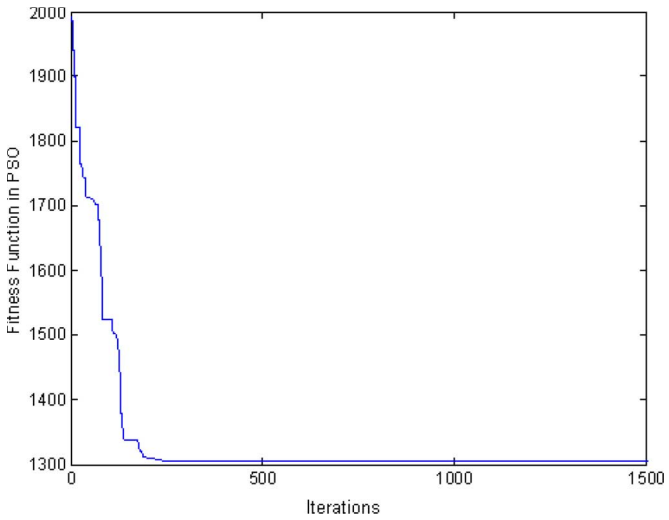


Fig. 10. Fitness function of the gbest particle with the PSO approach over iterations.

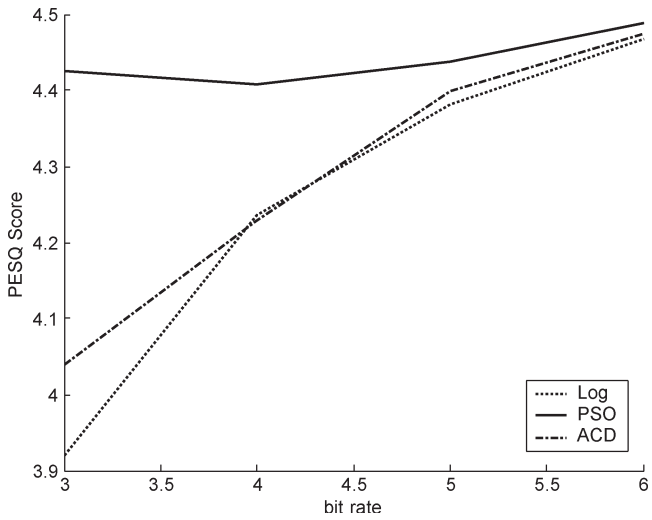


Fig. 12. PESQ score of the different quantizers.

TABLE I
COMPARISON OF SNR AND PESQ SCORES OF DIFFERENT QUANTIZERS FOR DIFFERENT BIT RATES ON THE TRAINING SPEECH DATA SAMPLE (FIG. 7)

Quantization Bit Rate		3	4	5	6
SNR (dB)	Log	7.78	13.39	19.54	25.81
	PSO	14.32	20.00	25.92	31.10
	ACD	8.33	14.26	20.36	26.33
PESQ Score	Log	3.92	4.24	4.38	4.47
	PSO	4.43	4.41	4.44	4.49
	ACD	4.04	4.23	4.40	4.48

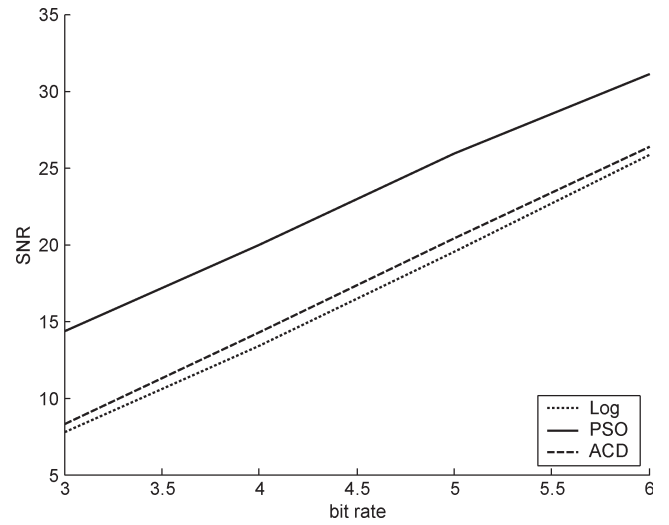


Fig. 11. SNR (in decibel) of the different quantizers.

The SNR and PESQ score for the training data (Fig. 8) are given in Table I and Figs. 11 and 12. In the following tables and figures, "Log" represents the logarithm quantizer, "PSO" represents the MLP-based quantizer developed using the PSO approach, and "ACD" represents the MLP-based quantizer developed using the ACD approach.

Based on the MLP 1 and MLP 2 weights obtained from the training data (Fig. 7), the MLP-based nonuniform quantizer is evaluated on the speech samples #1, #2, and #3, and their SNRs are given in Tables II–IV, respectively.

TABLE II
COMPARISON OF SNR AND PESQ SCORES OF DIFFERENT QUANTIZERS FOR DIFFERENT BIT RATES ON THE SPEECH SAMPLE #1 (FIG. 4)

Quantization Bit Rate		3	4	5	6
SNR (dB)	Log	5.82	12.30	18.11	23.07
	PSO	6.61	12.47	17.97	23.86
	ACD	6.30	12.98	18.55	23.35
PESQ Score	Log	1.83	2.22	2.93	3.34
	PSO	1.72	1.97	2.38	3.16
	ACD	1.79	2.27	2.93	3.35

TABLE III
COMPARISON OF SNR AND PESQ SCORES OF DIFFERENT QUANTIZERS FOR DIFFERENT BIT RATES ON THE SPEECH SAMPLE #2 (FIG. 5)

Quantization Bit Rate		3	4	5	6
SNR (dB)	Log	7.50	13.59	19.33	24.51
	PSO	9.00	15.01	19.63	25.78
	ACD	7.87	14.17	19.75	24.35
PESQ Score	Log	2.08	2.53	2.98	3.45
	PSO	1.61	2.06	2.77	3.11
	ACD	2.11	2.56	3.04	3.46

From the results, it is observed that the MLP-based quantizer designed using ACD or PSO has better performance than logarithm quantizer in SNR or PESQ score. Since PSO searches directly for SNR, it produces the highest SNRs; however, it does not guarantee high PESQ score (except the training data which might contain too few samples for evaluating subjective quality), and this proves that the objective quantization distortion like the sum squared error and SNR is not equivalent to subjective quality. While the ACD makes an improvement over the standard logarithm quantizer, it produces higher SNR as well as PESQ score than the logarithm quantizer for most speech samples.

TABLE IV
COMPARISON OF SNR AND PESQ SCORES OF DIFFERENT QUANTIZERS FOR DIFFERENT BIT RATES ON THE SPEECH SAMPLE #3 (FIG. 6)

Quantization Bit Rate		3	4	5	6
SNR (dB)	Log	7.60	13.27	19.39	25.98
	PSO	15.36	21.15	26.72	31.76
	ACD	8.26	14.03	20.14	26.51
PESQ Score	Log	2.14	2.40	2.73	3.07
	PSO	2.15	2.49	2.80	3.11
	ACD	2.16	2.44	2.92	3.11

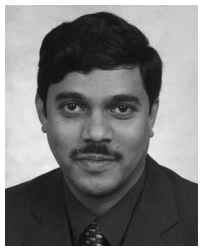
VII. CONCLUSION

This paper has presented the design of companding nonuniform optimal scalar quantizers for speech coding. Two new approaches for the optimal design based on ACDs and PSO have been shown using feedforward neural networks. An objective measure has been applied in the PSO- and ACD-based design approaches, and the performances of both designs are evaluated on two measures (SNR and PESQ). The perceptual quality of the ACD and PSO-based neural-network coding evaluated by the ITU's PESQ standard are maximized for a range of bits. It is observed that the approximate dynamic-based nonuniform quantizer design approach yields higher PESQ ratings compared with the PSO-based approach which yields higher SNRs.

Future work is to implement a mathematical model for evaluating subjective quality of speech coding, like PESQ, as the utility function in ACD or fitness function in PSO.

REFERENCES

[1] M. Akune, R. M. Heddle, and K. Akagiri, "Super bit mapping: Psychoacoustically optimized digital recording," presented at the 93rd Conv. Audio Eng. Soc., San Francisco, CA, 1992.
[2] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall, 1984, ch. 4.
[3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.
[4] P. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control*, D. A. White and D. A. Sofge, Eds. New York: Van Nostrand Reinhold, 1992, pp. 493–525.
[5] J.-W. Park, R. G. Harley, and G. K. Venayagamoorthy, "Adaptive-critic-based optimal neurocontrol for synchronous generators in a power system using MLP/RBF neural networks," *IEEE Trans. Ind. Appl.*, vol. 39, no. 5, pp. 1529–1540, Sep./Oct. 2003.
[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, Nov. 27–Dec. 1, 1995, vol. 4, pp. 1942–1948.
[7] G. K. Venayagamoorthy and S. Doctor, "Navigation of mobile sensors using PSO and embedded PSO in a fuzzy logic controller," in *Proc. 39th IEEE IAS Annu. Meeting*, Oct. 3–7, 2004, vol. 2, pp. 1200–1206.
[8] V. G. Gudise and G. K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks," in *Proc. Swarm Intell. Symp.*, Indianapolis, IN, Apr. 24–26, 2003, pp. 110–117.
[9] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.
[10] R. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proc. Congr. Evol. Comput.*, May 2001, vol. 1, pp. 81–86.
[11] *Perceptual Evaluation of Speech Quality (PESQ)*, ITU-T Rec. P.862, Feb. 2001.



Ganesh Kumar Venayagamoorthy (S'91–M'97–SM'02) received the B.Eng. degree (with first class honors) in electrical and electronics engineering from Abubakar Tafawa Balewa University, Bauchi, Nigeria, in 1994 and the M.Sc.Eng. and Ph.D. degrees in electrical engineering from the University of Natal, Durban, South Africa, in 1999 and 2002, respectively.

He was a Senior Lecturer with Durban Institute of Technology, South Africa, prior to joining the University of Missouri, Rolla (UMR), in May 2002. He is currently an Associate Professor of Electrical and Computer Engineering and the Director of the Real-Time Power and Intelligent Systems Laboratory at UMR. His research interests are in computational intelligence, signal processing, evolvable hardware, and power systems. He has published over 200 papers in refereed journals and international conferences proceedings.

Dr. Venayagamoorthy is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and a Senior Member of the South African Institute of Electrical Engineers (SAIEE). He is also a member of the International Neural Network Society (INNS), The Institution of Engineering and Technology, U.K., and the American Society for Engineering Education. He is currently the IEEE St. Louis Computational Intelligence Society (CIS) and IAS Chapter Chairs, the Chair of the Task Force on Intelligent Control Systems, the Secretary of the Intelligent Systems subcommittee and the Vice-Chair of the Student Meeting Activities subcommittee of the IEEE Power Engineering Society, and the Chair of the IEEE CIS Task Force on Power System Applications. He has organized and chaired several panel, invited, and regular sessions, and tutorials at international conferences and workshops. He was a recipient of the 2004 NSF CAREER Award, the 2006 IEEE Power Engineering Society Walter Fee Outstanding Young Engineer Award, the 2006 IEEE St. Louis Section Outstanding Section Member Award, the 2005 IEEE Industry Applications Society (IAS) Outstanding Young Member Award, the 2005 SAIEE Young Achievers Award, the 2004 IEEE St. Louis Section Outstanding Young Engineer Award, the 2003 INNS Young Investigator Award, the 2001 IEEE CIS Walter Karplus Summer Research Award, five prize papers from the IEEE IAS and IEEE CIS, a 2006 UMR School of Engineering Teaching Excellence Award, and a 2005 UMR Faculty Excellence Award. He is listed in the 2007 edition of *Who's Who in America.*

Wenwei Zha (S'05) received the Bachelor's degree in electrical engineering from Tsinghua University, Beijing, China, in 2004 and the Master of Science degree in computer engineering from the University of Missouri, Rolla (UMR), in 2006.

He was a Graduate Assistant and a Teaching Assistant from August 2004 to July 2005 and from August 2005 to May 2006, respectively, with UMR, where he was also a member of the Real-Time Power and Intelligent Systems Laboratory. He is currently an ASIC Design Engineer with ESS Tech Inc., Fremont, CA.