ELSEVIER

Signal Processing 81 (2001) 1513–1523

SIGNAL PROCESSING

www.elsevier.nl/locate/sigpro

Vector quantization based on genetic simulated annealing

Hsiang-Cheh Huang[a], Jeng-Shyang Pan[b,*], Zhe-Ming Lu[c], Sheng-He Sun[c], Hsueh-Ming Hang[a]

[a]*Department of Electronic Engineering, National Chiao-Tung University, Taiwan*
[b]*Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, 415 Chien-Kung Road, Kaohsiung, Taiwan*
[c]*Department of Automatic Test and Control, Harbin Institute of Technology, Harbin, China*

Received 18 November 1999; received in revised form 25 February 2001

Abstract

Genetic algorithm (GA) has been successfully applied to codebook design for vector quantization (VQ). However, most conventional GA-based codebook design methods need long runtime because candidate solutions must be fine tuned by LBG. In this paper, a partition-based GA is applied to codebook design, which is referred to as genetic vector quantization (GVQ). In addition, simulated annealing (SA) algorithm is also used in GVQ to get more promising results and the corresponding method is referred to as GSAVQ. Both GVQ and GSAVQ use the linear scaling technique during the calculation of objective functions and use special crossover and mutation operations in order to obtain better codebooks in much shorter CPU time. Experimental results show that both of them save more than 71–87% CPU time compared to LBG. For different codebook sizes, GVQ outperforms LBG by 1.1–2.1 dB in PSNR, and GSAVQ outperforms LBG by 1.2–2.2 dB in PSNR. In addition, GVQ and GSAVQ need a little longer CPU time than, the maximum decent (MD) algorithm, but they outperform MD by 0.2–0.5 dB in PSNR.

Keywords: Vector quantization; Codebook design; Genetic algorithm; Simulated annealing

1. Introduction

As an effective method for data compression, VQ has been successfully used in speech coding and image compression. The k-dimensional, N-level vector quantizer is defined as a mapping from a k-dimensional Euclidean space R^k into a certain finite set $C = \{c_1, c_2, \ldots, c_N\}$. The quantizer is completely described by the codebook C together with the partitioned set consisting of subspaces of R^k, $S = \{s_1, s_2, \ldots, s_N\}$, and the mapping function $Q(\bullet)$:

$$Q(X) = c_i, \quad \text{if } X \in s_i. \tag{1}$$

The elements of the partitioned set S satisfy $\cup_{i=1}^{N} s_i = R^k$ and $s_i \cap s_j = \Phi$, if $i \neq j$. If the squared Euclidean distortion measure is used, the output of the vector quantizer is the index i of the codeword c_i which satisfies

$$i = \arg\min_{p} \sum_{l=1}^{k} (x^l - c_p^l)^2. \tag{2}$$

Codebook design is the key problem of VQ and the generated codebook has more effect on the

* Corresponding author. Tel.: +886-7-381-4526 ext. 5636; fax: +886-7-389-9382.
E-mail address: jspan@cc.kuas.edu.tw (J.-S. Pan).

0165-1684/01/$ - see front matter
PII: S0165-1684(01)00048-2

Nomenclature

VQ	vector quantization	$a_{li}^{j}(t)$	the ith gene of unit j for individual l in population $P(t)$
GA	genetic algorithm	$X_{a_{li}^{j}(t)}$	the training vector whose label is $a_{li}^{j}(t)$in the chromosome
GVQ	genetic vector quantization	$Y_l^r(t)$	the rth codeword for individual l in population $P(t)$
GSAVQ	genetic simulated annealing vector quantization	$C_l(t)$	the corresponding codebook of individual $A_l(t)$
SA	simulated annealing	$D_l(t)$	the overall distortion of individual$A_l(t)$
LBG	conventional codebook design algorithm	$D_{\min}(t)$	the overall distortion of the best individual$A_{\min}(t)$
Unit j	the jth partitioned set or the jth cluster	$S_l(t)$	the inverse of the overall distortion $D_l(t)$
$P(t)$	population of the tth generation	$f_l(t)$	the fitness function of individual $A_l(t)$
$A_l(t)$	the lth individual for population $P(t)$	$p_l(t)$	the selected probability of individual $A_l(t)$
$A_{\min}(t)$	the best individual that has minimal overall distortion in population $P(t)$		
$n_l^j(t)$	the number of genes that belong to unit j for individual l in population $P(t)$		

compression performance. The traditional codebook design method—LBG algorithm [7] is affected by the initial codebook, often generates the local optimal codebook and needs intensive computation. Research efforts in codebook design have been concentrated in two directions: to generate a better codebook that approaches the global optimal solution, and to reduce the computational complexity.

To generate better codebooks, SA has been proposed in [13]. The SA method attempts to obtain a better codebook by shaking the codebook off the local valley in the hope that it will converge to another valley that gives less error. The SA method makes the LBG algorithm converge again and again to obtain a lower overall error. Thus, the computation time of SA is much longer in comparision with LBG. The stochastic relaxation approach [14] is also proposed to improve the codebook design. The basic idea of stochastic relaxation approach is to add some values to the codewords definitely for each iteration. In addition, deterministic annealing [12] has been used for codebook design recently. However, these methods need a great deal of time in order to obtain better codebooks.

To reduce the codebook generation time, many methods have appeared in the literatures. The subspace distortion method [11] attempts to reduce the computation time by reducing the dimension measure in the LBG algorithm. The pairwise nearest neighbor (PNN) algorithm [2] generates a codebook by merging nearest training vector clusters until the desired number of codewords is obtained. The codebooks generated by both methods are slightly degraded even though the computation time is reduced by several times.

Maximum decent algorithm [8] was also proposed for VQ codebook design. The algorithm begins with treating the training vector set as a global cluster. The algorithm generates the required number of clusters one by one using the maximum criterion until the desired number of codewords are obtained. Compared with the LBG algorithm, the codebook performance is improved and the computation time is substantially reduced. However, this algorithm can hardly obtain a global optimal codebook.

The aim of the codebook design is to find the best classification of training vectors. Given the number of the training vectors M and the number of

codewords N, codebook design problem is an NP-hard problem to classify M training vectors into N clusters. For larger M and N, a traditional search method can hardly find the global optimal classification. GA [4–6,10] is an efficient, parallel and near global optimum search method based on the ideas of nature selection and nature genetics. During the search processing, it can automatically achieve and accumulate the knowledge about the search space, and adaptively control the search process to approach the global optimal solution. However, the convergence speed of GA is a little slower because of its poor local optimum search ability. Genetic algorithm [1,9] has been used to generate better codebooks in recent years. Delport and Koschorreck proposed the partition-based GA codebook design algorithm [1] whose coding string is the codebook indices of the training data. The codebook vectors were also used as the coding string to design the codebook [9] which can be referred to as the codebook-based codebook design algorithm. However, previous GA-based codebook design algorithms have the same shortcoming, i.e., long runtime. This shortcoming is caused by the following two reasons: (1) all candidates are iterated by LBG, and (2) the crossover operation is not efficient. In order to improve the crossover operation, genetic algorithm with deterministic crossover [3] has been used to improve the effectiveness of the crossover operation, but this algorithm still needs much more CPU time than LBG algorithm. SA is a near global optimum search method based on the idea of physical annealing, whose operation object is not a group of approximate solutions like GA but a single approximate solution. The convergence speed of SA is higher than GA, but the performance of SA is affected by a lot of factors and it is not so easy to approach the global optimal solution. Because of this, we can make full use of the virtues of GA and SA and present a better codebook design method referred to as genetic simulated annealing vector quantization (GSAVQ). In the next section, both GVQ and GSAVQ algorithms are intensively discussed.

2. GVQ and GSAVQ algorithms

2.1. Genetic vector quantization

In this section, we will introduce a partition-based GVQ codebook design algorithm. Suppose that the number of training vectors, the codebook size and the vector dimension are M, N, and K, respectively. The label $i\,(i = 1,2,\ldots,M)$ of the training vector is viewed as a gene. A basic object for genetic operations is a classification of training vectors. The basic object is made up of N units, and each unit is made up of several labels that belong to this unit. N codewords are used to represent N units; here, a codeword is the centroid of the vectors whose indices belong to a certain unit. Suppose a population is made up of L individuals, then the population at the tth generation can be described as $P(t) = \{A_1(t), A_2(t), \ldots, A_L(t)\}$. Individual $A_l(t)(l = 1,2,\ldots,L)$ of population $P(t)$ can be illustrated as in Fig. 1. In Fig. 1, $n_l^j(t)(j = 1,2,\ldots,N)$ is the number of genes in unit j, and $\sum_{j=1}^{N} n_l^j(t) = M$. $(a_{li}^j(t)i = 1,2,\ldots,n_l^j(t))$ is the ith gene of unit j. If the training set can be described as $S = (X_1, X_2, \ldots, X_M)$, then $a_{li}^j(t)$ is the label of vector $X_{a_{li}^j(t)}$. Suppose the corresponding codebook of individual $A_l(t)$ can be described as $C_l(t) = \{Y_l^1(t), Y_l^2(t), \ldots, Y_l^N(t)\}$, then codeword $Y_l^r(t)(r = 1,2,\ldots,N)$ can be calculated by the following formula

$$Y_l^r(t) = \frac{\sum_{i=1}^{n_l^r(t)} X_{a_{li}^r(t)}}{n_l^r(t)}. \tag{3}$$

The main idea of GVQ is to find the training vector with maximum distance from the centroid (codeword) of a unit for one individual and then randomly move this training vector to the other unit or

$a_{l1}^1(t), a_{l2}^1(t)\cdots, a_{ln_l^1(t)}^1(t)$	$a_{l1}^2(t), a_{l2}^2(t)\cdots, a_{ln_l^2(t)}^2(t)$	.....	$a_{l1}^N(t), a_{l2}^N(t)\cdots, a_{ln_l^N(t)}^N(t)$

Fig. 1. Individual $A_l(t)$.

move this training vector to the other unit according to the condition of the second individual. The GVQ algorithm can be stated as follows:

Step 1: *Population initialization*. GA is strongly robust; so, how to generate the initial population has little effect on the codebook performance but may have some effect on the runtime. The population is required to include enough number of individuals to ensure the great diversity of individuals in the population. In this paper, the initial population $P(0)$ is made up of $L(L > 1)$ individuals, which are randomly generated, namely, $P(0) = \{A_1(0), A_2(0), \ldots, A_L(0)\}$.

Step 2: *Fitness function computation*. The overall distortion $D_l(t)$ of individual $A_l(t)$ can be expressed as

$$D_l(t) = \sum_{r=1}^{N} \sum_{i=1}^{n_l^r(t)} |X_{a_{li}^r(t)} - Y_l^r(t)|^2, \tag{4}$$

where $Y_l^r(t)$ is the rth codeword in codebook $C_l(t)$, $X_{a_{li}^r(t)}$ is the training vector whose label is $a_{li}^r(t)$, and $|X_{a_{li}^r(t)} - Y_l^r(t)|^2$ is the squared Euclidean distance between $X_{a_{li}^r(t)}$and $Y_l^r(t)$, i.e., the distortion of the input vector $X_{a_{li}^r(t)}$. Because the ultimate aim of VQ is to obtain the best classification that has the least overall distortion, the objective function $S_l(t)$ can be directly defined as the inverse of the overall distortion, namely, $S_l(t) = 1/D_l(t)$.

Fitness function is usually used to evaluate the fitness of an individual for the environment. In this paper, fitness function is used to evaluate the codebook performance. The more fitness the individual has, the higher performance its corresponding codebook has. Thus, fitness function can be defined as the objective function $S_l(t)$, as defined above. In order to restrain the probable misguided inclination of "super-individual" at the beginning of the search process and avoid converging too early, the linear scaling technique is used in this paper. Suppose the maximum objective function of Population $P(t)$ is $S_{\max}(t)$, and the average objective function of population $P(t)$ is $S_{\text{avg}}(t) = \sum_{l=1}^{L} S_l(t)/L$, then the fitness function $f_l(t)$ $(l = 1,2, \ldots, L)$ of individual $A_l(t)$ can be defined as

$$f_l(t) = \frac{S_{\text{avg}}(t)}{S_{\max}(t) - S_{\text{avg}}(t)}(S_l(t) + S_{\max}(t) - 2S_{\text{avg}}(t)). \tag{5}$$

Step 3: *Selection*. The individuals with more fitness ought to have more opportunity to be selected as the mother individuals of next generation. The selection probability $p_l(t)(l = 1,2, \ldots, L)$ of individual $A_l(t)$ can be defined as

$$p_l(t) = \frac{[f_l(t) - f_{\min}(t)]^2}{\sum_{l=1}^{L} [f_l(t) - f_{\min}(t)]^2}, \tag{6}$$

where $f_{\min}(t) = \min(f_1(t), f_2(t), \ldots, f_L(t))$. Then L pairs of mother individuals are selected according to the probability $p_l(t)(l = 1,2, \ldots, L)$.

Step 4: *Crossover*. Each pair of mother individuals generates a new individual by the crossover operation, thus L pairs generate a new population. As shown in Fig. 2, the crossover approach of mother individual 1 and mother individual 2 can be described as follows: (a) In individual 1, select a certain unit m randomly and find vector j in unit m which has maximum distortion. (b) In individual 2, find unit p including vector j and select another vector u in this unit randomly. (c) In individual 1, find unit n including vector u. (d) In individual 1, if $m \neq n$, push vector j from units m to n with certain crossover probability p_c. (e) Repeat the above operations for several times. The finally obtained individual 1 is the individual of child population. In this paper, the number of repeated times equals the population size.

Step 5: *Mutation*. In order to avoid the local optimum problem, the mutation operation is essential. As shown in Fig. 3, for each individual of the child population which is obtained by crossover operations, select two units randomly, for example, units m and n, move vector j in unit m which has the maximum distortion in unit n with certain probability p_m.

Step 6: *Termination*. Two termination criteria are used here. For the first one, the process is executed for fixed number of iterations and the best solution obtained is taken to be the optimal one. For the second one, the algorithm is terminated if no further improvement in the fitness value of the best solution is observed for five iterations, and the best solution is take as an optimal one. If one of the above termination criteria is satisfied, then the algorithm is terminated. Otherwise, go to Step 2.

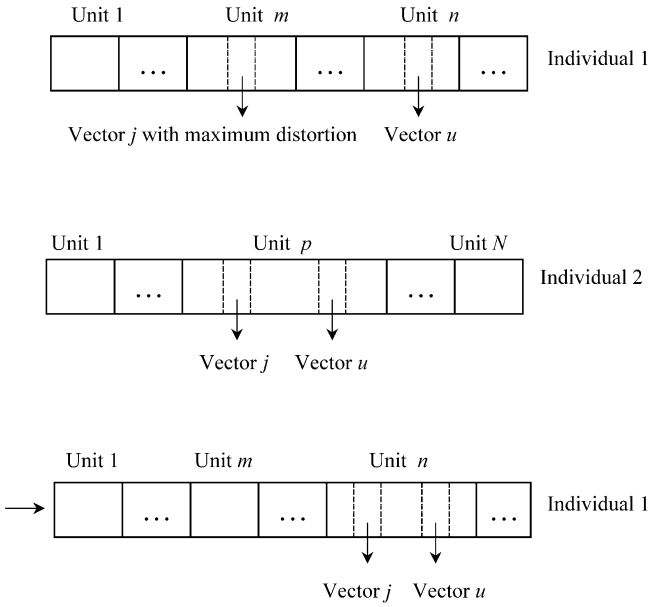

Fig. 2. Crossover approach of GVQ.

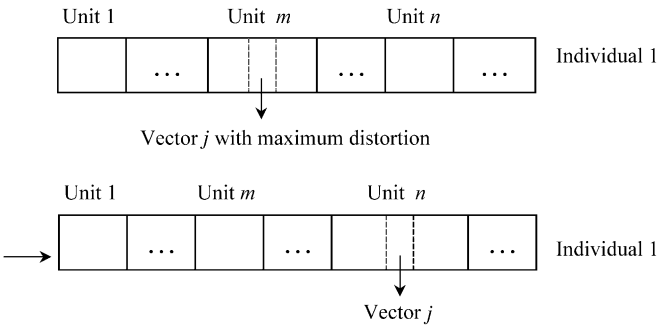

Fig. 3. Mutation approach of GVQ.

From above, we know that the proposed GVQ is partition-based, i.e., each solution is not a codebook but a partition of the training vectors. Secondly, the genetic operations are different from the conventional GA-based VQ. The conventional SA-based VQ and GA-based VQ often include LBG iterations in each iteration of SA or GA. However, in this paper, GLA iterations are not included in GA. Moreover, in the proposed semi-deterministic and semi-random crossover and mutation operations, the labels whose corresponding vectors have maximum distortion are moved in order to let the child solution approach the global optimal solution with more opportunities. In addition, no codeword search process is required in the calculation of the objective function. So the proposed GVQ is fast and efficient.

2.2. Genetic simulated annealing algorithm

In order to improve the local optimum search ability of GA and avoid the "premature phenomeno" of GA, SA is introduced in GVQ algorithm. SA is a global optimum search method based on the idea of physical annealing, whose operation object is not a group of approximate solutions like

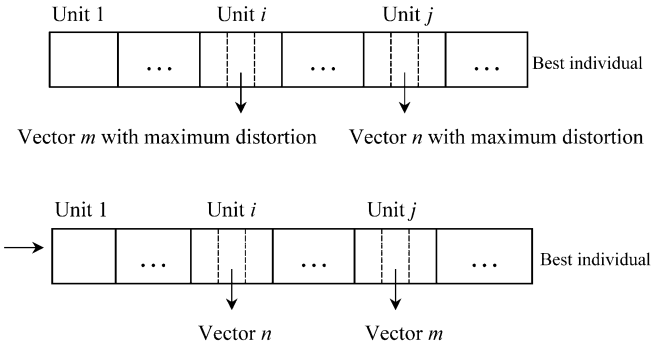

Fig. 4. Exchange approach of GSAVQ.

GA but a single approximate solution. The convergence speed of SA is higher than GA, but the initial temperature and the decreasing speed of the annealing temperature affect the performance of SA. If the initial temperature is high enough and the annealing temperature decreases slowly enough, the obtained solution approaches the global solution with probability one but the convergence speed is very low. On the other hand, SA is easy to fall into local optimum if the annealing temperature decreases quickly. If SA is used in the environment of a group of approximate solutions and made some modification, the performance of SA can be improved. The genetic simulated annealing vector quantization (GSAVQ) can be described as follows:

Step 1. Initialize parameters T_0 and threshold $\varepsilon(0 < \varepsilon < 1)$, set $t = 0$, $i = 0$, $D_{\min}(-1) = \infty$.

Step 2. Generate an intial population $P(0) = \{A_1(0), A_2(0), \ldots, A_L(0)\}(L > 1)$.

Step 3. Calculate the overall distortion for each individual in $P(t)$.

Step 4. In $P(t)$, find the best individual $A_{\min}(t)$ that has minimum overall distortion. If the overall distortion $D_{\min}(t)$ of $A_{\min}(t)$ satisfies $|(D_{\min}(t) - D_{\min}(t-1))|/D_{\min}(t) < \varepsilon$ or the iteration times t is larger than a given value, the algorithm is terminated with the corresponding codebook $C_{\min}(t) = \{Y^1_{\min}(t), Y^2_{\min}(t), \ldots, Y^N_{\min}(t)\}$ whose codeword $Y^r_{\min}(t)(r = 1,2,\ldots,N)$ can be calculated by

$$Y^r_{\min}(t) = \frac{\sum_{i=1}^{n^r_{\min}(t)} X_{a^r_{\min i}(t)}}{n^r_{\min}(t)}, \tag{7}$$

otherwise continue.

Step 5. Decrease the annealing temperature in a certain way, i.e. $T_{i+1} = T_i - \Delta T_i(\Delta T_i > 0)$, set $i = i + 1$.

Step 6. Generate a new individual $A'_{\min}(t)$ in the neighbourhood of individual $A_{\min}(t)$. As shown in Fig. 4, $A'_{\min}(t)$ is generated as follows: (1) select two units randomly, for example, units i and j; (2) find the vector which has maximum distortion in each unit, for example, vector m in unit i and vector n in unit j; (3) exchange vector m with vector n; (4) repeat the above operations for several times. Here, the number of repeated times is set to be the same as the population size.

Step 7. Compute the overall distortion$D'_{\min}(t)$ for $A'_{\min}(t)$ and $D_{\min}(t)$ for $A_{\min}(t)$, then calculate the difference $\Delta D(t) = D'_{\min}(t) - D_{\min}(t)$. If $\Delta D(t) < 0$, accept $A'_{\min}(t)$ as a new individual of $P(t)$, else if $\Delta D(t) \geqslant 0$, accept $A'_{\min}(t)$ as a new individual of $P(t)$ with probability $\exp(-\Delta D(t)/T_i)$.

Step 8. If $A'_{\min}(t)$ is accepted, add it into $P(t)$ as a new individual. Otherwise, randomly select an individual in $P(t)$ and add it into $P(t)$ as a new individual.

Step 9. For the previous population $P(t)$, generate a new population $P(t + 1)$ using genetic operations (steps 3–5 of GVQ). Remove the individual that has the worst performance from $P(t + 1)$. Set $t = t + 1$, go to step 3.

From above, we know that GSAVQ applies simulated annealing method to decide whether a training vector with maximum distance from the center (codeword) of a unit is suitable to be exchanged with the other training vector, and then combine with GVQ algorithm to get a better

codebook. Indeed, the incorporation of SA into GVQ can be viewed as the additional mutation operation besides genetic mutation operations, which can further reduce the opportunities of the algorithm getting into the local minimum. So GSAVQ can obtain better performance than GVQ in theory.

3. A concrete example

To explain the crossover and mutation operations of the proposed algorithms, we give a concrete example here. Suppose there are 16 two-dimensional training vectors, the aim is to generate four codewords. Thus $M = 16$, $N = 4$ and $K = 4$. Here, we assume the 16 training vectors to be $X_1 = (13,14)$, $X_2 = (7,8)$, $X_3 = (0,3)$, $X_4 = (2,7)$, $X_5 = (3,6)$, $X_6 = (1,7)$, $X_7 = (15,14)$, $X_8 = (2,11)$, $X_9 = (6,14)$, $X_{10} = (5,10)$, $X_{11} = (10,4)$, $X_{12} = (1,2)$, $X_{13} = (5,8)$, $X_{14} = (10,14)$, $X_{15} = (11,9)$ and $X_{16} = (15,12)$.

First, we explain how an individual corresponds to a codebook. Assume a certain individual can be expressed as $A_1 = \{(16,9,4,7,5),(2,10),(13,6,14,1,3,12),(8,11,15)\}$, with each element being the label of a training vector, e.g., the element '16' denotes the training vector X_{16}. The labels are classified into four units, and the numbers of elements in four units are $n_1^1 = 5$, $n_1^2 = 2$, $n_1^3 = 6$ and $n_1^4 = 3$, respectively. The codeword of a unit is the centroid of the training vectors whose labels belong to this unit, thus the individual A_1 corresponds to a codebook $C_1 = \{Y_1, Y_2, Y_3, Y_4\}$, where $Y_1 = (X_{16} + X_9 + X_4 + X_7 + X_5)/5 = (8,11)$, $Y_2 = (X_2 + X_{10})/2 = (6,9)$, $Y_3 = (X_{13} + X_6 + X_{14} + X_1 + X_3 + X_{12})/6 = (5,8)$, $Y_4 = (X_8 + X_{11} + X_{15})/3 = (8,8)$.

Then we turn to explain the crossover operation used in GVQ and GSAVQ. Assume two individuals $A_1 = \{(16,9,4,7,5),(2,10),(13,6,14,1,3,12),(8,11,15)\}$ and $A_2 = \{(2,4,8,5),(3,15,7),(11,14,1,3,12),(6,10,13,9)\}$ are selected to perform the crossover operation. Firstly, in individual A_1, select a certain unit m randomly and find vector j in unit m which has maximum distortion. Assume unit 1 is selected, i.e., $m = 1$, then vector $X_7(j = 7)$ has maximum distortion in unit $(16, 9, 4, 7, 5)$, i.e., $d(X_7, Y_1) = \max\{d(X_{16}, Y_1), d(X_9, Y_1), d(X_4, Y_1), d(X_7, Y_1), d(X_5, Y_1)\}$. Secondly, in individual A_2, find unit p including vector j and select another vector u in this unit randomly. Based on the above assumption, i.e., $j = 7$, then $p = 2$. And assume we select vector $X_3(u = 3)$ in unit 2. Thirdly, in individual A_1, find unit n including vector u. Based on the above assumptions, we know that vector X_3 is in unit 3 of individual A_1, i.e., $n = 3$. Fourthly, In individual A_1, if $m \neq n$, push vector j from unit m to unit n with certain crossover probability p_c. Based on the above assumptions, because $m = 1$, $n = 3$, $m \neq n$, then we can push vector X_7 from units 1 to 3 in individual A_1. After the above four steps, we can obtain new individual $A'_1 = \{(16,9,4,5),(2,10),(13,6,14,1,7,3,12),(8,11,15)\}$. Repeat the above four steps for several times, then we can obtain the final child individual A_1.

Finally, we turn to explain the mutation operation. Assume the individual $A_1 = \{(16,9,4,7,5),(2,10),(13,6,14,1,3,12),(8,11,15)\}$ is selected to perform the mutation operation. The operation is performed as follows: select two units randomly, for example, units 1 and 4, move vector X_7 in unit 1 which has the maximum distortion into unit 4 with certain probability p_m. If the move is performed, then the individual A_1 becomes another new individual $A'_1 = \{(16,9,4,5),(2,10),(13,6,14,1,3,12),(7,8,11,15)\}$.

4. Experimental results

To evaluate the efficiency of the proposed algorithms, GSAVQ and GVQ, accompanied with SA, MD and LBG, were coded in Visual C++ language and run on Pentium II computer. Two images, Lena and Peppers, with resolution 512×512 pixels, 8 bits per pixel, were used in this paper. The Lena image was used to generate the codebooks of different sizes with dimension 16 (4×4), the Peppers image was used to test the performance of the codebook. For LBG, the initial codebook is generated randomly and no speeding up methods are used in LBG and $\varepsilon = 0.001$. In the MD algorithm, the successive search method is used for searching the optimal partitioning hyperplane. For both GVQ and GSAVQ, the crossover probability p_c is 0.9 and the mutation probability p_m is 0.01, the population size is 40, the number of

Table 1
The CPU time(s) comparisons of LBG, SA, MD, GVQ and GSAVQ for different codebook sizes

Codebook size	LBG	SA	MD	GVQ	GSAVQ
64	132	3300	11	37	36
128	256	6656	23	59	54
256	485	12 271	45	97	90
512	965	23 374	89	162	123

Table 2
Performance (PSNR) comparisons of LBG, SA, MD, GVQ and GSAVQ for the image inside the training set

Codebook size	LBG (dB)	SA (dB)	MD (dB)	GVQ (dB)	GSAVQ (dB)
64	27.68	28.32	28.24	28.75	28.83
128	28.78	29.50	29.41	29.95	30.01
256	30.23	31.97	31.86	32.38	32.41
512	32.47	34.13	34.02	34.50	34.56

Table 3
Performance (PSNR) comparisons of LBG, SA, MD, GVQ and GSAVQ for the image outside the training set

Codebook size	LBG (dB)	SA (dB)	MD (dB)	GVQ (dB)	GSAVQ (dB)
64	26.58	27.44	27.35	27.65	27.70
128	27.81	28.67	28.54	28.85	28.87
256	29.33	31.27	31.20	31.48	31.52
512	31.13	33.09	33.01	33.30	33.35

iterations is limited to 150. For GSAVQ, the initial temperature T_0 is 50 and the temperature is decreased by 0.6% after each iteration until the number of iterations reaches 150, $\varepsilon = 0.001$. For SA [14], the initial temperature T_0 is 35 and the temperature is decreased by 1% after each iteration step until the number of iterations reaches 30. Experimental results are shown in Tables 1–4. As shown in Table 1, the five algorithms are compared in CPU time by generating different codebook sizes. Table 2 compares the five algorithms in PSNR for different codebook sizes using Lena as the test image. For the image outside the training set, Table 3 shows the performance of codebooks with different sizes generated by different algorithms. Fig. 5 shows the comparison of the test picture and pictures recovered by different algorithms. Fig. 6 shows the relationship of PSNR and computation time for codebook sizes 128 and 256, and because the CPU time of SA is nearly 25–27 times that of LBG, the relationship of PSNR and computation time of SA is excluded in Fig. 6. Compared with LBG method for codebook sizes 64, 128 and 256, GVQ and GSAVQ both save more than 71%, 77% and 80% CPU time, respectively. For codebook size 512, GVQ saves more than 83% CPU time and GSAVQ saves more than 87% CPU time compared with LBG. For different codebook sizes, the codebooks obtained by GVQ outperform those by LBG method by 1.1–2.1 dB in PSNR, and the codebooks obtained by GSAVQ outperform those by LBG method by 1.2–2.2 dB in PSNR. For different codebook sizes, the codebooks obtained by GVQ and GSAVQ outperform those by the SA method by 0.2–0.5 dB in PSNR. Thus the effectiveness of GVQ and GSAVQ is proven. Although MD algorithm can obtain the codebook in a much shorter time than GVQ and GSAVQ, its PSNR is lower than SA, GVQ and GSAVQ. In addition, we can see that the difference in the performance between GVQ and GSAVQ is slight after limited iteration times. However, if the iteration time is unlimited, the performance of GSAVQ will be much better than GVQ.

Table 4
Performance (MSE and time) comparisons of LBG, GVQ and GSAVQ for coding four-dimensional Guass–Markov sources with different codebook sizes

Codebook size	Performance (s)	LBG	GVQ	GSAVQ
8	Time	12.16	3.86	3.84
	MSE	0.679	0.543	0.539
16	Time	25.26	6.74	6.72
	MSE	0.342	0.277	0.276
32	Time	51.15	12.32	12.28
	MSE	0.169	0.151	0.147

In order to demonstrate the effectiveness of the proposed algorithms, we also made another

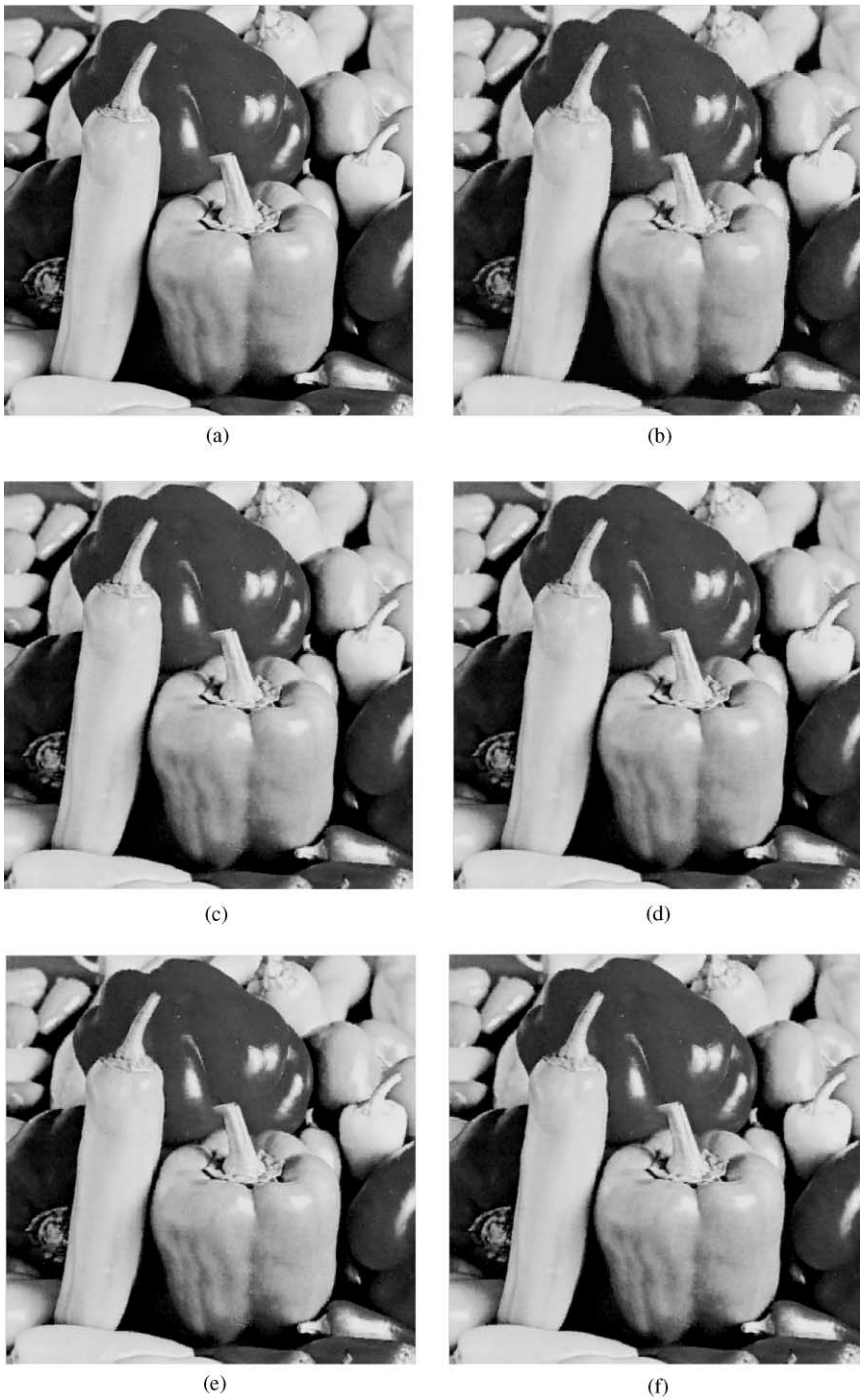

Fig. 5. Comparisons of pictures recovered by different algorithms for 256 codewords. (a) Original picture. (b) Picture recovered by LBG. (c) Picture recovered by SA. (d) Picture recovered by MD. (e) Picture recovered by GVQ. (f) Picture recovered by GSAVQ.

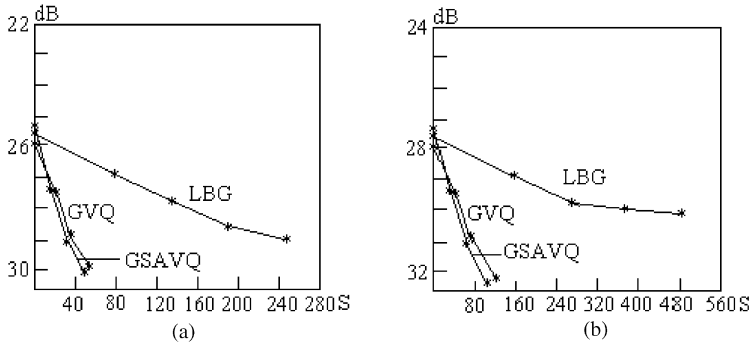

Fig. 6. Relationship between PSNR and CPU time for different codebook sizes for LBG, GVQ and GSAVQ algorithms (a) size 128 and (b) size 256.

experiment. In this experiment, the proposed algorithms together with the LBG algorithm were used to design VQ codebooks using 1000 four-dimensional Guass–Markov vectors with unit variance and zero mean and $\sigma = 0.5$ in each dimension. Each algorithm generates three codebooks with sizes 8, 16 and 32. The experimental results (MSE and time) are shown in Table 4. From Table 4, we can see that the proposed algorithms can obtain better performance in a shorter time compared to LBG.

5. Summary and conclusion

In this paper, two methods for codebook design are presented. These two algorithms are genetic vector quantization (GVQ) and genetic simulated annealing vector quantization (GSAVQ). For both algorithms, the label of the training vector is viewed as a gene and the basic object for genetic operations is a classification of training vectors. The basic object is made up of a certain number (equals to codebook size) of units, and each unit is made up of several labels that belong to this unit. A certain number of codewords are used to represent these units; here a codeword is the centroid of a unit. In order to restrain the probably misguided inclination of "super-individual" at the beginning of the search process and avoid converging too early, the linear scaling technique is used in both algorithms. Both GVQ and GSAVQ make full use of the near global optimum ability of genetic algorithm (GA), and GSAVQ also utilizes the virtues of simulated annealing (SA) in order to avoid the "premature phenomenon" of GA and obtain a near global optimum codebook in a much shorter time. Experimental results show that GVQ and GSAVQ both have better performance and need shorter computer time than SA and LBG algorithms, and GSAVQ is superior to GVQ. In addition, although GVQ and GSAVQ need a little longer CPU time than maximum decent (MD) algorithm, they outperform MD in PSNR.

References

[1] V. Delport, M. Koschorreck, Genetic algorithm for codebook design in vector quantization, Electron. Lett. 31 (2) (1995) 84–85.
[2] W.H. Equitz, A new vector quantization clustering algorithm, IEEE Trans. Acoust. Speech Signal Process 37 (10) (1989) 1568–1575.
[3] P. Franti, Genetic algorithm with deterministic crossover for vector quantization, Pattern Recognition Lett. 21 (2000) 61–68.
[4] M. Gen, R. Cheng, Genetic Algorithms and Engineering Design, Wiley, New York, 1997.
[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley Publishing Company, Reading, MA, 1989.
[6] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, MI, 1975.
[7] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design, IEEE Trans. Commun. 28 (1) (1980) 84–95.
[8] C.K. Ma, C.K. Chan, Maximum descent method for image vector quantization, Electron. Lett. 27 (12) (1991) 1772–1773.

[9] J.S. Pan, F.R. McInnes, M.A. Jack, VQ codebook design using genetic algorithms, Electron. Lett. 31 (17) (1995) 1418–1419.

[10] J.S. Pan, F.R. McInnes, M.A. Jack, Application of parallel genetic algorithm and property of multiple global optima to VQ codevector index assignment, Electron. Lett. 32 (4) (1996) 296–297.

[11] L.M. Po, Novel subspace distortion measurement for efficient implementation of image vector quantizer, Electron. Lett. 26 (29) (1990) 480–482.

[12] K. Rose, Deterministic annealing for clustering, compression, classification, regression and related optimization problems, Proc. IEEE 86 (1998) 2210–2239.

[13] J. Vaisey, A. Gersho, Simulated annealing and codebook design, Proceedings ICASSP'88, 1988, pp. 1176–1179.

[14] K. Zeger, A. Gersho, Stochastic relaxation algorithm for improved vector quantizer design, Electron. Lett. 25 (14) (1989) 896–898.