Globally Optimal Vector Quantizer Design by Stochastic Relaxation

Kenneth Zeger, Jacques Vaisey, and Allen Gersho, *Fellow, IEEE*

***Abstract*—This paper presents a unified formulation and study of vector quantizer design methods that couple stochastic relaxation (SR) techniques with the generalized Lloyd algorithm. Two new SR techniques are investigated and compared: simulated annealing (SA), and a reduced-complexity approach that modifies the traditional acceptance criterion for simulated annealing to an unconditional acceptance of perturbations. It is shown that four existing techniques all fit into a general methodology for vector quantizer design aimed at finding a globally optimal solution. Comparisons of each algorithms' performance when quantizing Gauss–Markov processes, speech, and image sources are given. The SA method is guaranteed to perform in a globally optimal manner, and the SR technique gives empirical results equivalent to those of SA. Both techniques result in significantly better performance than that obtained with the generalized Lloyd algorithm.**

I. Introduction

ONE of the most challenging problems in the theory of source coding is to design vector quantizers that introduce as little average distortion as possible. Vector quantization (VQ) is a source coding technique that approximates blocks (or vectors) of input data by one of a finite number of prestored vectors in a codebook [1], [2]. The effectiveness of the codebook design is the primary challenge in constructing vector quantizers. A widely used design approach defines the data source through the use of a finite training set, and uses an iterative procedure called the generalized Lloyd algorithm (GLA) [3] to produce a codebook giving a locally minimal average distortion. The algorithm is essentially the same as the k-means algorithm used in pattern recognition. The performance of a quantizer produced by the GLA is often considerably inferior to that of a globally optimal quantizer, although it has certain "locally optimal" properties. An open question in communication theory is how to effectively design quantizer codebooks that are globally optimal in performance.

One approach to solving this problem is to consider design algorithms that do not share the deterministic nature of the GLA. The GLA is a descent algorithm, i.e., at each iteration in the algorithm a positive improvement in performance is achieved over the previous iteration. The GLA thus has the advantage that it converges to a final codebook relatively quickly; however, since codebook design is an off-line operation, its complexity does not affect the operation or performance of the coder, and the computational limit is set only by the available computing power.

Several powerful optimization techniques that introduce elements of randomness into algorithms have been investigated in the past. Rastrigin [4] discusses "random search" techniques for obtaining arbitrarily accurate solutions to optimization problems. The basic idea is to randomly perturb the state of the system at each iteration of the algorithm and determine the resulting change in performance. If the performance increases, the perturbation is accepted; however, the change is withdrawn if the performance decreases. Rastrigan also analyzes several variations on this theme. His "blind search" technique consists of repetitively applying the perturbations described above. If the perturbations consist of choosing the next state in an arbitrary manner from the current state, then the globally optimal solution will be found eventually. However, the computational requirements for this approach are unreasonable, since finding the global optimum may take an enormous amount of time. A more efficient "blind search" method limits the proposed perturbations to some small neighborhood of the current state, with neighborhoods diminishing in size as the algorithm progresses. The "best trial" search examines a collection of potential perturbations at each iteration, and selects the one that yields the best performance. The "search with linear scaling" technique accepts a perturbation if it increases performance, otherwise it accepts a perturbation in the opposite direction. This last method, unlike the previous ones, allows decreases in system performance at each iteration, since a step in the opposite direction may degrade the performance. An application of these ideas to antenna array design was made by Widrow and McCool [5], who compare a "linear random search (LRS)" method resembling Rastrigin's linear scaling technique to two steepest descent algorithms.

In most of Rastrigin's techniques, the value of the objective function must decrease with every additional iteration of the minimization (the linear scaling method is an

Manuscript received June 8, 1989; revised June 11, 1990. This work was supported in part by the National Science Foundation, Bell Northern Research, Rockwell International, Bell Communications Research, Inc., and the University of California MICRO Program. Portions of this paper were presented at ICASSP, New York, NY, Apr. 11-14, 1988.

K. Zeger is with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822.

J. Vaisey is with the School of Engineering Science, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6.

A. Gersho is with the Communications Research Laboratory, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106.

IEEE Log Number 9104885.

example of where this restriction does not necessarily hold). To require a monotonic decreasing objective function is artificial and highly restrictive since there is no sound reason to always forbid perturbations that decrease performance. If a random search allows temporary decreases in an objective function with nonzero probability, then the algorithm is in the class of techniques called stochastic relaxation (SR) [6].

One important SR approach is simulated annealing (SA), a technique which unconditionally accepts perturbations that increase performance, and probabilistically accepts perturbations that decrease performance, with the probability depending on the amount of decrease. Under very general conditions, SA can be guaranteed to yield globally optimal solutions to combinatorial problems [7], and has recently been used to solve a variety of engineering problems (e.g., [8]–[16]). Direct use of SA has been attempted in several specific VQ design procedures (referred to as VQ-SA) [14]–[16] as well as some effective reduced complexity SR techniques [3], [17].

In this paper, we present a unified formulation of vector quantizer design using stochastic relaxation algorithms (referred to as VQ-SR). General design procedures are initially presented that combine the benefits of stochastic algorithms with those of the GLA. It is shown that the VQ-SR algorithms divide in a natural way into two disjoint classes, those with encoder perturbations and those with decoder perturbations. Within these classes, there is another relevant subdivision among the algorithms, those that strictly adhere to the philosophy of SA, and those that utilize reduced complexity SR methods. It is shown that each of the existing VQ-SR design procedures falls into one of these four subclasses, and that some potentially promising variations on these have yet to be studied.

Experimental comparisons are made between the various VQ-SR techniques, and it is concluded that the reduced complexity VQ-SR methods generally perform as well or better in practice as the full complexity VQ-SA methods, which are theoretically guaranteed to yield globally optimal performance. This paper describes the details of these applications of SR to VQ design and presents experimental evidence indicating that substantial performance increases can be achieved with these new techniques over the standard GLA. In some cases, the performance improvement using VQ-SR is equivalent to doubling the codebook size using the standard GLA. This implies a transmission rate reduction of 1 b per vector index, a significant savings in many low-rate speech and image coding applications. Equivalently, without sacrificing performance, a reduction in computational complexity by more than a factor of two is obtained in this way, an important attribute for real-time applications.

The paper is organized as follows. Section II introduces the general properties of SA and VQ, defining terms and highlighting the features that are relevant to optimization problems. Section III presents a general formulation of VQ-SA design techniques, and Section IV gives simplified VQ-SR design algorithms. Section V then presents numerical results that compare the performances of the algorithms on various speech, image, and Gauss–Markov sources.

II. Simulated Annealing and Vector Quantization

Simulated annealing [7], [19] is a technique that relates combinatorial optimization problems to the physical process of cooling a molten metal. In this analogy, a random initial state (or set of parameter values) in the optimization corresponds to the melted state, and local minima in the objective function relate to the formation of different crystal structures once the metal freezes into a solid. In metallurgy, a technique called annealing is often used, where a very fine crystal structure is obtained through gradual cooling. Too rapid a cooling schedule results in a coarse structure, similar to a poor local minimum in an optimization. The correspondence between the two processes is created through the Metropolis algorithm [20], a procedure based on a mathematical description of the behavior of a collection of atoms under constant temperature.

In an optimization problem, the goal is to find, in a multidimensional input space, a parameter (or state) vector $S_{\min}$ that minimizes some nonnegative real-valued objective function. In its most direct form, SA iteratively searches through the input space in a nondeterministic manner and attempts to minimize an energy function, E, which is usually, but not necessarily (e.g., see [8]), equal to the objective function. The essential idea behind SA is to add randomness to the search for the global minimum of the energy function, allowing the algorithm to "pull" itself out of local minima.

The system's state at the mth iteration of an SA algorithm S_m is a random vector that specifies all of the parameters in the optimization. A perturbation function, π, maps a system's state to another state according to some probability law. In each step of SA, the current system state S_m is tentatively altered to $\pi(S_m)$ by a trial random perturbation, or change, and the resulting energy increase, $\Delta E_m \doteq E[\pi(S_m)] - E(S_m)$, is calculated. When no confusion results we may omit the subscript and simply write ΔE. If ΔE_m is negative, then the system moves to the new state, $\pi(S_m)$; however, when ΔE_m is positive, a probabilistic decision is made whether or not to implement the trial perturbation in the state. The state change is accepted with a probability that decreases exponentially with the size of $\Delta E_m / T_m$, where for each m, T_m is a nonnegative real number called the temperature. It is assumed that the sequence $\{T_m\}$ is monotonic nonincreasing and that $\lim_{m \to \infty} T_m = 0$. More explicitly, if for each m we define the random variable Φ_m as

$$\Phi_m = \begin{cases} 1 & \text{if } S_{m+1} = \pi(S_m) \\ 0 & \text{if } S_{m+1} = S_m \end{cases} \tag{1}$$

then $\Phi_m = 1$ if and only if the perturbation at step m is accepted, and the probability of accepting the change is given by

$$\Pr(\Phi_m = 1) = \begin{cases} e^{-\Delta E_m/T_m} & \text{if } \Delta E_m \geq 0 \\ 1 & \text{if } \Delta E_m < 0 \end{cases} \tag{2}$$

which we call the SA acceptance rule. If the change is not kept, then the system simply remains at S_m. When the temperature is large, almost every proposed state change will be implemented. A small T_m implies that almost no perturbations that increase the energy will be allowed. In the limiting case when $T_m = 0$, the algorithm is no longer able to escape from local minima. The ability of the system to move to states of higher energy gives the SA algorithm its power to avoid nonoptimal local minima.

If a sufficient number of iterations, or trial perturbations, are made at a constant temperature, then the system eventually attains a state known as thermal equilibrium, where the relative probability that the system is in any given state versus any other stabilizes. In fact, if the SA acceptance rule is used, then at thermal equilibrium the relative probabilities between two states S_1 and S_2 at iteration m is

$$\frac{\Pr(S_m = S_1)}{\Pr(S_m = S_2)} = e^{[E(S_2) - E(S_1)]T_m}. \tag{3}$$

Thus, when thermal equilibrium is reached at temperatures close to zero, the lowest energy (the global optimum) is much more probable than any other. However, it is not feasible to directly run the algorithm at a low temperature since the number of perturbations needed to reach thermal equilibrium from a given initial condition increases dramatically as the temperature is lowered [7].

The fastest way to reach thermal equilibrium at low temperatures is to start with a high temperature where thermal equilibrium can be reached quickly [7]. This initial high temperature "melts" the system, randomizing the state since almost all the proposed changes will be accepted. The temperature is held constant until thermal equilibrium is approached, at which time the temperature at the next iteration is reduced to the next value in a predetermined monotonically decreasing sequence of temperatures, $\{\hat{T}_n\}$, called the cooling schedule or simply, schedule. Note that the schedule $\{\hat{T}_n\}$ is a subsequence of the temperature sequence $\{T_n\}$ and contains exactly one copy of each distinct temperature. The algorithm is then run (for many iterations) at the new temperature until thermal equilibrium is restored, a process whose duration is dependent on the amount of temperature drop. If the cooling is done too quickly, then it may take a very long time to regain equilibrium after each temperature drop. With a straightforward application of Markov chains, it can be shown that for sufficiently slow temperature decreases and simple constraints on the allowed perturbation functions, the probability that the system state has not reached the global optimum $S_{\min}$ can be made arbitrarily close to zero [7]. A sufficient constraint on π is that every system state be "reachable" in the sense that for a nonzero temperature, any given state can eventually be obtained from any other state with nonzero probability as a consequence of perturbations. Connors and Kumar [18] showed that under a mild set of assumptions, a global optimum is attained (with probability approaching one as $n \to \infty$) if and only if for a certain constant d^* the series $\sum_{n=1}^{\infty} \exp(-d^*/\hat{T}_n)$ diverges. The schedule

$$\hat{T}_n = \frac{C}{\log(n+1)} \tag{4}$$

meets this requirement if and only if $C \geq d^*$, where $\hat{T}_n$ is the nth temperature in the schedule. In practice, this schedule is very slow and Kirkpatrick [19] recommends using a "suboptimal" exponential schedule of the form

$$\hat{T}_n = \hat{T}_0 \cdot K^n \tag{5}$$

where K is a positive constant less than unity (equal to 0.9 in [19]). The particular choices of the permutation function and the temperature schedule are the two major factors that affect the SA performance for specific applications, and are often chosen by experimentation.

A vector quantizer Q is a mapping of Euclidean k-dimensional space $\boldsymbol{R}^k$ into a finite collection of points in $\boldsymbol{R}^k$, called a codebook. A vector quantizer approximates an input source vector from $\boldsymbol{R}^k$ as accurately as possible by one of the vectors in a predetermined codebook. A vector quantizer can be decomposed into an encoder C and a decoder D. The encoder is specified by a partition of the input space $\boldsymbol{R}^k$ into a finite collection of regions $\{R_1, \cdots, R_N\}$, where for each $\boldsymbol{x} \in R_i$, $C(\boldsymbol{x}) = i$. The decoder is determined by a codebook, $\{y_1, \cdots, y_N\}$, where $D(i) = y_i$, for each i.

The quantity to be minimized (the objective function) in VQ is the average distortion per sample

$$e \doteq E[d(X, Q(X))] \tag{6}$$

where d is a distortion function, usually chosen as the squared-error function given by

$$d(X, Y) \doteq \frac{1}{k} \|X - Y\|^2. \tag{7}$$

Two necessary conditions on the encoder and decoder of a vector quantizer are known, in order to obtain a minimum average squared-error distortion [2]. These conditions, the nearest neighbor (NN) for the partition condition, and the centroid condition for the codevectors, are utilized in the paper by Linde *et al.* [3] to generalize the Lloyd algorithm [21] for quantizer design from scalar sources to vector sources. The resulting algorithm is called the generalized Lloyd algorithm (GLA), and is an iterative descent technique that produces a sequence of vector quantizers with monotonically decreasing average distortions for a given source. A description of the GLA is given in Fig. 1, where M denotes the number of training vectors, N the number of codevectors in the codebook, $\boldsymbol{x}_i$ the ith training vector, and at the mth iteration $y_j^{(m)}$ denotes the jth codevector, $R_j^{(m)}$ the jth partition region, and D_m the distortion.

```
(1) Codevector Initialization: $y_1^{(1)}, \cdots y_N^{(1)}$.
        $m = 1$
        $D_0 = \infty$

(2) Nearest Neighbor Repartition ($1 \le i \le M$):
        $j = argmin\{ \|x_i - y_l\| : 1 \le l \le N \}$.
        Put $x_i \to R_j^{(m)}$.
        $D_m = D_m + \|x_i - y_j^{(m)}\|^2$.

(3) Stopping Criterion:
        If $(D_{m-1} - D_m)/D_m < \varepsilon$ Stop.
        $m = m + 1$

(4) Centroid Computation ($1 \le i \le N$):
        $y_i^{(m)} = 1/|R_i^{(m)}| \sum_{x_i \in R_i} x_i$

(5) Goto (2)
```

Fig. 1. The GLA for codebook design.

Each iteration of the GLA (called a Lloyd iteration) can be viewed as a nearest neighbor (NN) repartitioning followed by centroid computations. Thus, after each Lloyd iteration is completed, the resulting quantizer satisfies the necessary centroid condition and produces a MSE less than or equal to that obtained in the previous iteration. For a finite training set, the GLA converges to a locally optimal quantizer in a finite number of iterations; however, the monotonically decreasing nature of the objective function may trap the algorithm in a locally optimal solution, perhaps very different from the global optimum [22].

In a modification of the GLA applied to image compression, Vaisey and Gersho [14] obtained improved codebooks by combining SA with the Lloyd iteration. The perturbation was implemented by randomly moving the training vectors between neighboring partition regions until thermal equilibrium is reached. In addition, a single GLA iteration was incorporated each time equilibrium was achieved to further improve convergence. More recently, Flanagan *et al.* [15] investigated essentially the same approach as in [14], but used a complexity reduction technique for computing ΔE after each training vector perturbation, and used no GLA iterations. The technique in [14] is discussed in detail in Section III.

In their paper describing the GLA, Linde *et al.* [3] also present a modified version of their well-known fixed-point technique for VQ design, with the goal of obtaining the globally optimal solution. Their idea was to corrupt the source by additive independent noise, whose marginal distribution is such that only one locally optimum quantizer exists for it. As the noise power is gradually reduced, the modified algorithm converges to a codebook that is generally better than the fixed-point method. This approach is similar to the technique in [14] except that perturbations are added to the source unconditionally in [3]. Since the change in energy plays no role in this modified technique, the approach is not strictly a form of SA, but instead belongs to the broader class of SR algorithms.

Further applications of SA and SR to VQ design have been given by Cetin and Weerackody [16] and Zeger and Gersho [17], respectively. In [17], results comparable to those in [14] were obtained with a significantly reduced computational complexity (presented in detail in Section IV).

III. Quantizer Design Using SA

As discussed in Section II, quantizer design can be viewed as an optimization problem, where the objective is to find the encoder and decoder that give the best performance in coding the training set. Since the GLA decreases the distortion at each iteration, the algorithm can easily get stuck in poor local minima. Questions that need to be addressed are whether or not we truly end up in a poor local minimum with the GLA, how much improvement is obtained using SA, and whether the full power of SA is needed to obtain this performance improvement. In the following, we present a general framework for applying SA to VQ design.

To apply SA to VQ design, a definition is needed for the system state. The state can be specified by an encoding and decoding rule, or, more specifically, a partition and a codebook. This description, however, leads to cumbersome and inefficient implementations. Since an optimal quantizer satisfies both the NN and centroid rules, a more efficient alternative representation of the state can be made by assuming one of these rules always holds. Thus, there are two natural ways in which this state can be conveniently viewed: 1) as a specific partition of the training set (in which case the centroid condition is assumed to always be satisfied) or 2) as the set of codevectors (in which case the NN condition is assumed). In each case, we present a convenient formulation of the perturbation operation.

In the first case, the perturbation is implemented by corrupting the input source by an additive noise component prior to each NN repartitioning. Here we regard the perturbation π as a probabilistic mapping that distorts input vectors, and define the "perturbed" quantizer coder C^* as the composition

$$C^* \doteq C \circ \pi. \tag{8}$$

By following the input perturbation π with NN partitioning, a perturbation is effectively performed on the quantizer partition associated with the original input. This method does not sacrifice any generality since any perturbation of a quantizer partition can be realized by choosing an appropriate mapping π.

In the second case, a perturbation applies noise to the reconstructed vectors ("jittering" the codevectors). The perturbation mapping π is again regarded as a function from $\boldsymbol{R}^k$ to $\boldsymbol{R}^k$ that maps each old codevector into a new codevector, and the perturbed quantizer decoder D^* is defined as

$$D^* \doteq \pi \circ D. \tag{9}$$

We refer to the perturbations in these two cases as encoder perturbations and decoder perturbations, respectively. A block diagram of the encoder and decoder perturbations is shown in Fig. 2. By a perturbed quantizer Q^* we mean a quantizer with either encoder or decoder perturbations. Thus, either

$$Q^* = D \circ C^* \quad \text{or} \quad Q^* = D^* \circ C. \tag{10}$$

Note that since every state is reachable, SA will yield globally optimal quantizers in both cases.

The issue of computational complexity is of great importance, since it is often the limiting factor in quantizer design algorithms. The most demanding computational task is the calculation of the energy change resulting from a trial perturbation. In all of the applications of SA to VQ design, the energy is chosen to equal the objective function, which is the mean-square distortion

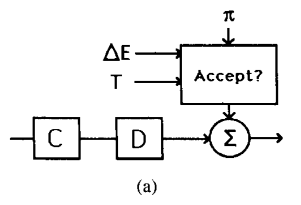
$$E = \sum_{R_i} \sum_{x \in R_i} d(x, y_i). \tag{11}$$

A block diagram of the general computation of ΔE for VQ is shown in Fig. 3, where each training vector x_i passes through both the perturbed and unperturbed quantizers to find its contribution, ΔE_i to ΔE.

There is a fundamental difference between the two perturbation methods in terms of the number of computations required to determine ΔE. For encoder perturbations, the centroid condition is assumed, so the centroids of the partition regions affected by a perturbation must be recomputed in order to calculate ΔE. However, this calculation is not difficult if one imposes the restriction that each perturbation affect the encoding of only one training vector. In this case, a minor modification of the old centroids will produce the new ones (as noted in [15]). On the other hand, the computation of ΔE is generally not computationally efficient in the case of decoder perturbations. The problem is that every training vector has a possibility of being assigned to a new cluster in the NN partition whenever any one codevector changes slightly in value. Thus, an entire repartitioning must be performed following every decoder perturbation, a procedure that is too cumbersome in many practical situations, though it may have value for the design of small size codebooks with small training sets. In Section V, we introduce a reduced complexity algorithm that efficiently implements a decoder perturbation. In the remainder of this section, we concentrate on an SA implementation with encoder perturbations.

A. Encoder Perturbation

In applying SA to the codebook design problem at the encoder, we seek a way to perturb the partition in a simple manner. In this case, the state is defined as the quantizer partition and our task is accomplished by moving a single training vector from one cluster to another, changing the centroids of the two clusters involved. Thus, the encoder



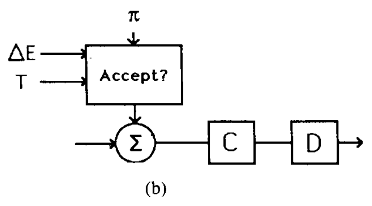


Fig. 2. The perturbation strategies. (a) Decoder perturbation. (b) Encoder perturbation.

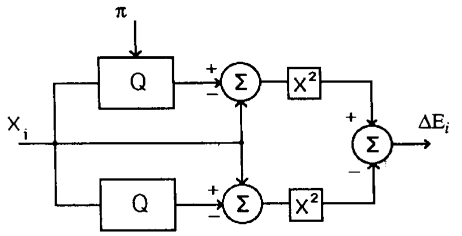


Fig. 3. The generation of ΔE. Each training vector x_i is run through both the perturbed and unperturbed quantizers to obtain the partial distortion increment.

perturbation SA algorithm, closely based on the method in [19], consists of picking a training vector from a target cluster and then randomly switching it to a destination cluster chosen from one of the N possible clusters.

We next define an important general class of perturbation functions by specifying their probability distributions. For any set S let $|S|$ denote the number of elements in S. For each training vector x, let $A(x)$ denote some subset of the codebook. The uniform encoder perturbation π is then defined by the probability distribution

$$\Pr[\pi(x) = y] = \begin{cases} |A(x)|^{-1} & \text{if } y \in A(x) \\ 0 & \text{if } y \notin A(x) \end{cases} \tag{12}$$

where y represents any vector in $\boldsymbol{R}^k$. That is, each training vector can be perturbed to one of a predetermined set of different partition regions with equal probabilities. The set of allowable regions to which a given training vector can be moved is the "neighborhood" of that vector. For convenience of implementation, we include the additional requirement that each $A(x)$ consist of the $\hat{N}$ nearest codevectors (based on Euclidean distance) to x, for some integer $\hat{N}$.

If $\hat{N} = N$, then each pick and switch is made with all choices being equally likely, and the perturbation function can be written as $\pi(x) = y_i$, where the y_i's are the codevectors, and i is a uniformly distributed random integer between 1 and N. In this case, as in [15] we calculate ΔE as

$$\Delta E = \sum_{x \in R_t - \{x_s\}} \frac{d(x, y_t^*)}{|R_t| - 1} + \sum_{x \in R_d \cup \{x_s\}} \frac{d(x, y_d^*)}{|R_d| + 1} - \sum_{x \in R_t} \frac{d(x, y_t)}{|R_t|} - \sum_{x \in R_d} \frac{d(x, y_d)}{|R_d|} \tag{13}$$

where x_s indicates the training vector to be switched, R_t and R_d denote the target and destination partition regions before the perturbation, y_t and y_d are the centroids of the target and destination clusters, and y_t^* and y_d^* are the centroids of the two clusters after the perturbation. The new centroids can be calculated easily since they are related to y_t and y_d according to

$$y_t^* = \frac{|R_t|y_t - x_s}{|R_t| - 1} \quad \text{and } y_d^* = \frac{|R_d|y_d + x_s}{|R_d| + 1} \tag{14}$$

Perturbations are then accepted according to (2). At each temperature $\hat{T}_n$, a fixed number L of switches is attempted and then the system is assumed to have reached thermal equilibrium. The design algorithm terminates when the number of accepted switches after L attempts in less than some threshold. When this condition occurs the system is declared to be frozen.

The above approach should lead to a good solution; however, since many switches must be attempted at each temperature, the number of calculations required is extremely large. The amount of computation required can be appreciated when one examines the "chip assignment" problem discussed in [19]. In this case, the task is to place 5000 circuits on one of two VLSI chips with the objective of minimizing the number of connections and routing on the chips. Kirkpatrick requires 5000 $\times$ 10 accepted switches (moving a circuit from one chip to the other) before it is assumed that thermal equilibrium has been reached. Roughly following his example, we should need $N_T \times \hat{N} \times 10$ accepted switches at each temperature; this is approximately equal to 10^7 when it is desired to design a codebook of 256 vectors from a training set of size 8192 and $\hat{N} = N$, a task that takes a very long time.

Since many switches may not be accepted, a better method to reduce the amount of computation is to require that $\hat{N} << N$. The perturbation function is now taken as $\pi(x) = \hat{y}_i(x)$, where the $\hat{y}_i$'s are the closest $\hat{N}$ codevectors to x, and i is a uniformly distributed random integer between 1 and $\hat{N}$.

We present an efficient technique (about 20% faster) called SA-C, that combines the GLA and SA procedures by following each series of L SA switch attempts by one iteration of the GLA, where, as in [19], we set L to be equal to ten times the number of training vectors. The nearest neighbors are computed after every L switches. From the perspective of the GLA, the new algorithm allows the distortion to increase between each GLA iteration, hopefully leading to a descent toward a better local minimum. These modifications allow the algorithm to converge to a good solution much faster than the basic SA approach implemented in [15], when the number of codevectors is large.

The size of $\hat{N}$ seriously affects the amount of required computation and the final codebook. A large $\hat{N}$ implies that most of the perturbations will be rejected at low temperatures where large energy changes are unlikely. To reach thermal equilibrium faster, we thus monitor the number of accepted perturbations for the farthest away of the $\hat{N}$ codevectors at each iteration (of L trials). The value of $\hat{N}$ is decreased by one if none of these switches are accepted ($\hat{N}$ is never dropped below a minimum value of two). Also, the use of relatively small initial values of $\hat{N}$ occasionally limits the possible performance improvement. Certain sources have most of their energy concentrated in a small region of the input space, and a random selection of initial codevectors will generally assign many codevectors to this region—perhaps many more than is desirable. If all of these vectors are nearest neighbors to each other, then our perturbation strategy will never move one of these vectors to other regions of the space, and we will be trapped in a local minimum. To solve this problem, a procedure was implemented when the lowest distortion cluster is deleted after every high temperature iteration (where "high" is heuristically chosen to be say the first 50% of the temperatures in the schedule). The deleted clusters are then replaced by "splitting" the cluster giving the highest contribution to E into two pieces. Splitting is a standard technique for handling empty clusters in the GLA, and a sample splitting procedure is described in [3].

Iterations of the SA-C algorithm are run one after the other at the current temperature until either the distortion between GLA iterations changes by less than 0.1%, or ten iterations have occurred since the last temperature drop. At this point, the temperature is decreased to the next value as specified by the schedule. A flowchart describing this algorithm is shown in Fig. 4.

The last remaining set of parameters to determine is the annealing schedule. In specifying this schedule, it is important that the temperature not drop too quickly, since this may cause the system to be far from thermal equilibrium at a low temperature. By experimentation, we have found that good performance is obtained when the initial temperature causes 15% to 25% of the switches to be accepted, and the schedule is of the form given in (5). When less than 0.1% of the switches are accepted at a given temperature, we declare the system to be frozen and run the GLA until the distortion converges to finish the design.

B. Decoder Perturbation

SA can be applied to VQ design by perturbing the quantizer decoder and representing the state by the codebook. Each iteration consists of three steps: applying a perturbation to the codebook; determining whether the perturbation is acceptable; and repartitioning the training set according to the NN rule. The perturbation is done by

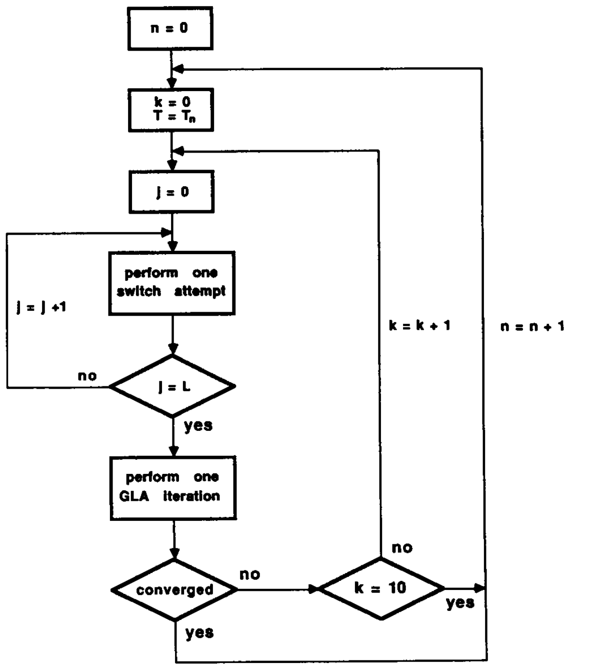


Fig. 4. The flowchart for the SA-C algorithm.

randomly selecting a codevector and altering its value in a probabilistic manner (e.g., by adding zero-mean vector-valued noise, whose component variances are directly proportional to the temperature). To determine whether a perturbation is accepted, the resulting energy change ΔE is calculated and the SA acceptance rule is applied using the current temperature.

For very low temperatures, perturbations will usually be accepted only when energy reductions result. In this case, codevectors will migrate towards the partition cell centroids, the optimal codevector choice for a given partition. In this way, the centroid condition will be approached with increasing accuracy as the system cools. Ideally, the single codevector perturbations and subsequent repartitions should be repeated at a fixed temperature until thermal equilibrium is reached, at which time the temperature can be lowered. If the temperature is reduced at a sufficiently slow rate, the probability that the energy function will not be globally optimal can be made arbitrarily small.

As was mentioned earlier in this section, computational complexity in the calculation of ΔE is a serious problem with this technique, and we have thus not implemented this algorithm; however, these problems may be eliminated by modifying the standard SA approach in a manner that allows an easily implementable algorithm. Reduced complexity algorithms are the subject of the next section.

IV. Reduced Complexity Quantizer Design by SR

As discussed previously, two major disadvantages of using a SA approach for VQ design are the complexity of computing the quantity ΔE and the slowness of reaching thermal equilibrium. For both the encoder and decoder perturbation methods, it is possible to avoid the difficulties associated with SA by introducing a SR design algorithm that introduces major complexity reduction techniques. No globally optimal asymptotic convergence theorem analogous to one for SA is known by the authors for this SR application and its main justification is empirical success. The approach offers a much simplified version of complicated SA algorithms and achieves comparable (usually slightly better) results in significantly less time.

In our reduced complexity SR algorithms, we assume the following rules at each iteration:

R1) Accept every proposed perturbation.
R2) Simultaneously either perturb all encoder or all decoder parameters (but not both).
R3) Perform a repartitioning and centroid computation.

The first requirement effectively bypasses the selector blocks in Fig. 2 and eliminates the need to calculate ΔE, the second speeds up the algorithm by making many changes at once, and the third necessitates a Lloyd iteration at the end of every perturbation. In addition, we impose the constraint $T_m = \hat{T}_m$, for all m, so that the temperature sequence and cooling schedule are identical. This implies that the temperature sequence is strictly decreasing at every iteration, and that the temperature is not kept constant for many iterations in the hopes of approaching thermal equilibrium.

It is important to point out how the above technique differs from the GLA and VQ-SA methods. The GLA consists only of R3) above, and thus the reduced complexity SR algorithm is equivalent to taking the usual GLA and perturbing the result unconditionally after each Lloyd iteration; as the temperature decreases, the behavior approaches that of the GLA.

The perturbation function π that we have chosen to use is one that adds an independent, uniformly distributed zero-mean noise process ξ defined on $\boldsymbol{R}^k$, to its input. Thus from (8) and (9) we get

$$\pi(X) = X + \xi \tag{15}$$

$$C^*(X) = C(X + \xi) \tag{16}$$

$$D^*(X) = D(X) + \xi \tag{17}$$

for the encoder and decoder perturbations, respectively. Each component of the noise vector has variance σ_ξ^2. These processes can be viewed pictorially as in Fig. 2. Other distributions (e.g., Gaussian) can be used for the noise process; however, the results are not very sensitive to this choice.

A. Reduced Complexity Encoder Perturbation

An example of a design approach that adds noise to the training set to improve the ultimate performance has been proposed by Linde *et al.* [3], where their strategy is to add a decreasing amount of Gaussian noise to the training set at each iteration of the GLA. The initial noise level is set high enough to completely dominate the training data. Their SR technique does not strictly fit the model presented, as it violates condition R3), since adding noise to a training vector induces a new encoder mapping. When the same perturbed training vectors are subsequently used to compute the cluster centroids, R3) is violated since a true computation of the centroids based on the unaltered training set is not performed.

In our approach to the reduced complexity encoder design, called the SR-C algorithm, we perturb the entire training set according to (16) to alter the partition, and then calculate the cluster centroids using the uncorrupted training vectors associated with each region. The noise added at the mth iteration is controlled by the following temperature schedule:

$$T_m = \sigma_x^2 \left(1 - \frac{m}{I}\right)^p \tag{18}$$

where I defines the number of iterations to be run. The initial temperature is identically equal to σ_x^2 (the variance of the training set components) and a value of $p = 3$ was found to give the best performance. Two other types of schedules of the form

$$T_m = \frac{\sigma_x^2}{(m+1)^p} \quad \text{and} \quad T_m = \sigma_x^2 \alpha^m \tag{19}$$

were investigated; however, (18) was found to give the best overall performance.

The heuristic justification of why this approach works can be appreciated by realizing that adding noise to the training vectors can only affect the distortion in a restricted fashion. If each component of the noise vector ξ added to the training vector x, has variance σ_ξ^2, and if the system energy is defined as in (11), then, on the average, the total energy contributed by cluster i at the mth iteration will increase by

$$\begin{aligned}\overline{\Delta E_m^i} &= \frac{1}{k} E_\xi \left[\sum_{x \in R_i} \left(\|x + \xi - y_i\|^2 - \|x - y_i\|^2\right)\right] \\ &= |R_i| \sigma_\xi^2.\end{aligned}$$

The initial high levels of noise essentially randomize the state; however, as the noise is reduced, the amount of energy that can be added also decreases, making it more difficult for the algorithm to leave deep minima in a single step. On the other hand, shallow local minima will not confine the state, and since the added noise goes to zero, it will be much more probable that the state will be in a deep minimum of the energy. This approach is closer in spirit to SA than the SR approach in [3], since the energy calculated after each iteration is independent of the noise that was added.

B. Reduced Complexity Decoder Perturbation

A reduced complexity SR algorithm for the decoder perturbation, called the SR-D algorithm, can be realized by perturbing every codevector according to (17) at each iteration. In other words, the usual GLA iteration is modified by adding a noise component to the codevectors following each centroid computation. This procedure can be written as an extra step in Fig. 1 as

(4.5) Codevector Jiggling ($1 \le i \le N$):

$$y_i^{(m)} = y_i^{(m)} + \xi_i(T_m).$$

The noise variance is again dictated by the temperature, which is gradually reduced as in (18), except that σ_x^2 now indicates the variance of the codevector components. The heuristic justification for this algorithm is essentially the same as that for the simplified encoder perturbation. In practice, the run times associated with this SR algorithm are significant reductions over those of the more elaborate SA version given in Section III-A, and yet the performance is usually indistinguishable or superior.

V. Experimental Results

We will now present the results obtained when we used our algorithms to design codebooks for several different sources. Our quoted execution times are based on those obtained with a 4-MIPS SUN 3/260 machine using a floating point accelerator.

The first source we considered was obtained by extracting 8192 nonoverlapping 16-dimensional vectors (corresponding to 4 × 4 blocks) from eight 512 × 512 monochrome training images with each pixel amplitude quantized to 8 b. The pixel means were calculated for each vector, uniformly quantized to 3 b, and then subtracted from the vector components to form the training set. These means were subtracted in order to allow us to use the codebooks to quantize images with reasonable quality. Four codebooks of size 32, 64, 128, and 256 were built using this training data. The second source examined was obtained from a segment of human speech sampled at a rate of 8 kHz, and partitioned into 4096 4-dimensional vectors and 4096 2-dimensional vectors. In the first case, we designed five codebooks of size 8, 16, 32, 64, and 128 vectors, while in the second case we built five codebooks of size 16, 32, 64, 128, and 256. Finally, we examined two cases for each of three first-order Gauss–Markov sources with correlation coefficients of 0, 0.5, and 0.9, respectively. These sources were generated using the equation $x_i = \alpha x_{i-1} + w_i$ to derive the value of the ith sample, where w_i is a sample of independent white Gaussian noise. In the first case, we blocked 8192 samples of the source into 4096 2-dimensional training vectors, and designed codebooks of size 16, 32, 64, 128, and 256. In the second we blocked 4096 samples into 1024 4-dimensional training vectors and designed codebooks of size 4, 8, 16, 32, and 64 vectors.

We designed the codebooks for each of the above sources using four algorithms: the GLA; the encoder perturbation algorithm (SA-C) with the initial value of $\hat{N}$ set

TABLE I
GLA PERFORMANCE FOR THE IMAGE SOURCE. THE AVERAGE AND PEAK VALUES WERE CALCULATED USING TEN DIFFERENT INITIAL CONDITIONS

Codebook Size	Ave. PSNR (dB)	Peak PSNR (dB)
32	26.92	26.92
64	27.65	27.65
128	28.34	28.36
256	29.11	29.12

TABLE II
THE GLA PERFORMANCE FOR THE SPEECH SOURCE. THE AVERAGE AND PEAK VALUES WERE CALCULATED USING TEN DIFFERENT INITIAL CONDITIONS

Vector Dim.	Codebook Size	Ave. SNR (dB)	Peak SNR (dB)
	8	7.29	7.30
	16	9.92	9.93
4	32	12.50	12.54
	64	14.50	14.62
	128	16.10	16.49
	16	13.48	13.49
	32	16.35	16.38
2	64	18.23	18.39
	128	21.44	21.81
	256	24.05	24.51

to be equal to min $(N/3, 10)$; the reduced complexity encoder perturbation algorithm (SR-C); and the reduced complexity decoder perturbation algorithm (SR-D). Each one of the last two algorithms was executed for the same number of iterations. We note that the SA-C algorithm is computationally intensive because of the requirements that ΔE be calculated after every perturbation and that thermal equilibrium be approached. Thus, in order to achieve good performance, the SA-C algorithm must be run much longer than the other two algorithms. However, it is also true that all of our temperature schedules have been set conservatively, and that nearly equivalent performance can be attained in most cases.

The baseline performance for each source was computed by running the GLA with ten different randomly selected initial conditions (to obtain an adequate indication of average performance). The average and the best GLA results are shown in Table I for the image data, Table II for the speech data, and Table III for the Gauss-Markov source. The performances of the SR algorithms compared with the best GLA performances are shown in Figs. 5-7. As an additional comparison, the GLA results when the initial VQ codebooks are designed using a splitting algorithm [3] (denoted "split" in the figures) are also given. The splitting algorithm can outperform the randomly initialized GLA at times, but its performance is quite inconsistent and generally yields significantly worse performance than the SR algorithms. In the case of the image source, the number given in the table is the peak signal-to-noise ratio (PSNR), the most popular measure of quality in image coding, while the SNR is used for the

TABLE III
GLA PERFORMANCE FOR THE FIRST-ORDER GAUSS-MARKOV SOURCE. THE SOURCE DATA POINTS WERE DERIVED USING THE EQUATION $x_i = \alpha x_{i-1} + w_i$, WHERE w_i IS A SAMPLE OF INDEPENDENT WGN, AND THE AVERAGE AND PEAK VALUES WERE CALCULATED USING TEN DIFFERENT INITIAL CONDITIONS

α	Vector Dim.	Codebook Size	Ave. SNR (dB)	Peak SNR (dB)
		4	2.07	2.07
		8	3.54	3.55
	4	16	4.94	4.98
		32	6.45	6.47
		64	8.19	8.21
0				
		16	9.66	9.66
		32	12.44	12.45
	2	64	15.35	15.38
		128	18.22	18.25
		256	20.73	20.73
		4	3.30	3.30
		8	4.58	4.58
	4	16	6.09	6.10
		32	7.46	7.49
		64	9.28	9.31
0.5				
		16	10.30	10.37
		32	13.09	13.12
	2	64	16.05	16.11
		128	18.77	18.89
		256	21.59	21.74
		4	6.46	6.48
		8	8.57	8.58
	4	16	10.35	10.40
		32	11.86	11.89
		64	13.52	13.55
0.9				
		16	13.63	13.64
		32	16.34	16.34
	2	64	18.94	19.13
		128	21.56	21.67
		256	24.95	25.01

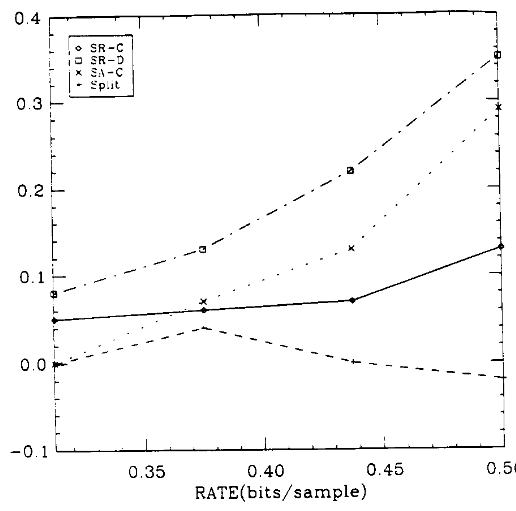
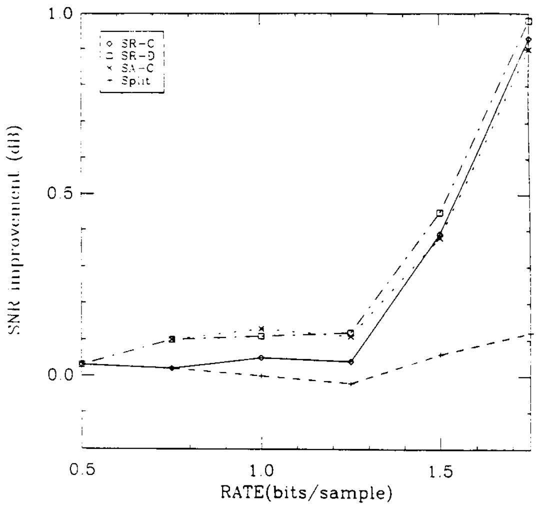


Fig. 5. The improvement obtained with SR on the image source.

other sources. The PSNR and SNR are defined as

$$\text{PSNR} = 10 \log [255^2/E] \quad \text{and} \quad \text{SNR} = 10 \log [P_s/E] \tag{20}$$



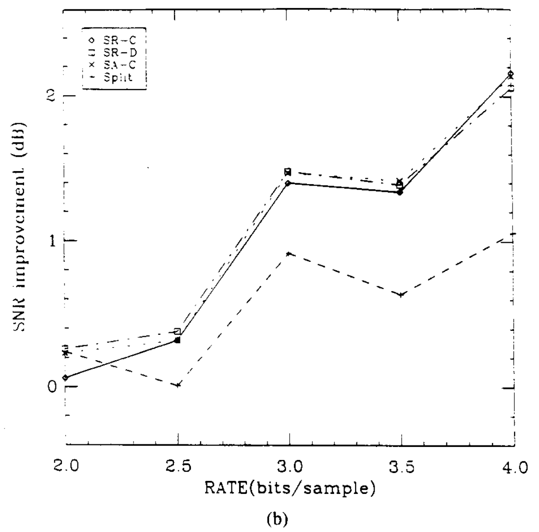
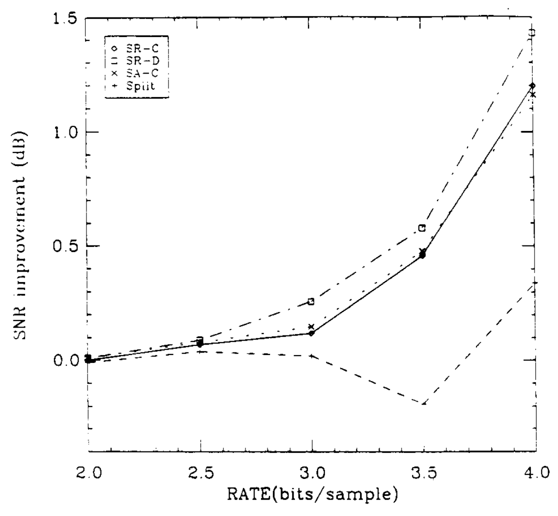


Fig. 6. The improvement obtained with SR on the speech source: (a) $k = 4$, (b) $k = 2$.



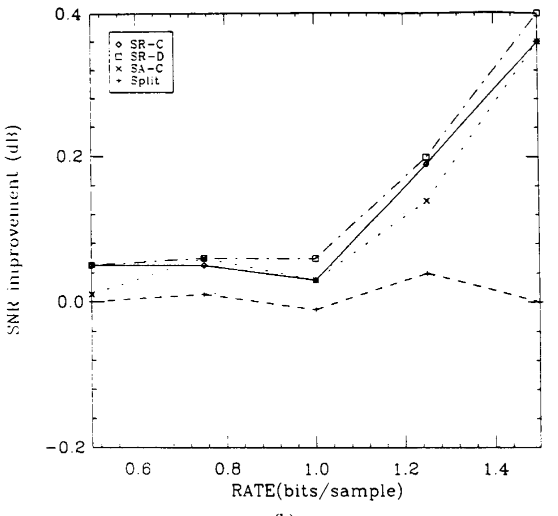


Fig. 7. The improvement obtained with SR on the Gauss–Markov source. The source data points were derived using the equation $x_i = \alpha x_{i-1} + w_i$, where w_i is a sample of independent WGN. (a) $k = 4$, $\alpha = 0$. (b) $k = 2$, $\alpha = 0$. *(Continued on next page.)*

where P_s is the signal power. We next present the SR results for each of the sources.

A. Image Source

The first design technique tested after running the GLA was the SA-C encoder perturbation. For our design of the codebooks, we follow (5) and use a schedule given by $\hat{T}_n = \hat{T}_0 \cdot (0.6)^n$, with the initial temperature causing approximately 25% of the switches to be accepted in the first iteration. Each iteration of the algorithm consists of $L =$ 81 920 switch attempts followed by an iteration of the GLA. The above schedule has been chosen heuristically; however, neither finer temperature divisions nor larger values of $\hat{T}_0$ improved the results significantly. Slightly improved performance is possible if the size of $\hat{N}$ is increased; however, the amount of computation grows rapidly with this parameter, since it then takes much longer to reach thermal equilibrium at each temperature. The SR-C and SR-D algorithms used the temperature schedule given in (18), with $I = 200$.

The PSNR's improvements obtained when using the SR techniques to design the codebooks are plotted in Fig. 5. From the figure, we see that each of the algorithms gives approximately the same performance. In each case, the improvement is significant, and in the 256 codevector case the improvement is approximately equal to that obtained by doubling the desired codebook size to 512 and using the GLA for the design. The entire design procedure in

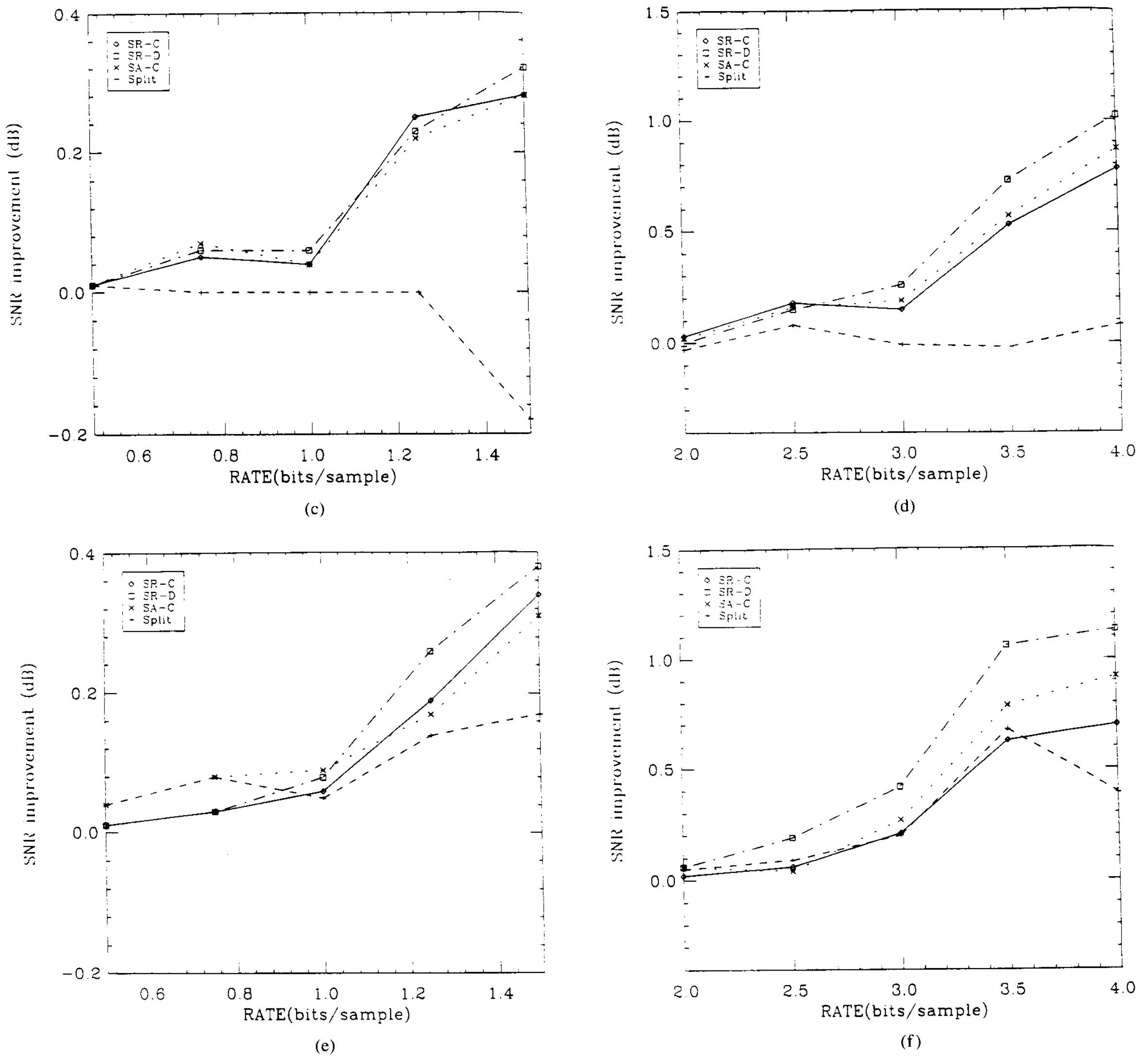


Fig. 7. (*Continued.*) (c) $k = 4$, $\alpha = 0.5$. (d) $k = 2$, $\alpha = 0.5$. (e) $k = 4$, $\alpha = 0.9$. (f) $k = 2$, $\alpha = 0.9$.

the 256 codevector case required approximately 450 min of CPU time for the SA-C algorithm and 340 and 190 min for the SR-C and SR-D algorithms, respectively. The GLA operating on the same training set typically needed 70 min to converge to a stabilized codebook (i.e., a true local minimum of the distortion). We note that the result obtained by running five iterations of the SA-C algorithm at zero temperature, implying that no perturbations that increase the energy will be accepted, seems to work almost as well when the full schedule is used, giving a PSNR of 29.3 dB. This last point indicates to us that the global minimum is probably not very deep and that there are many local minima with distortions close to the global. As in the previous example, the GLA terminates in a poor local minimum because it drops the energy in a very restricted fashion, making substantial changes to the trial codevectors only during the first couple of iterations.

The relative performance of these codebooks in the coding of an image (called "Lena") from outside of the training set is shown in Fig. 8. The SA-C image is seen to be superior in the high-detail regions, and the amount of "blockiness" in the neighborhood of the edges has been reduced. These high detail regions are identified by calculating the pixel variances of each 4 × 4 block. Thus, since we wish to examine the improvement in the high and low detail vectors separately, we group the vectors into high and low detail classes and calculate separate PSNR's. Through subjective evaluation, we found that a threshold of 144 provided a good separation between the two types of vectors, and the resulting numbers are shown

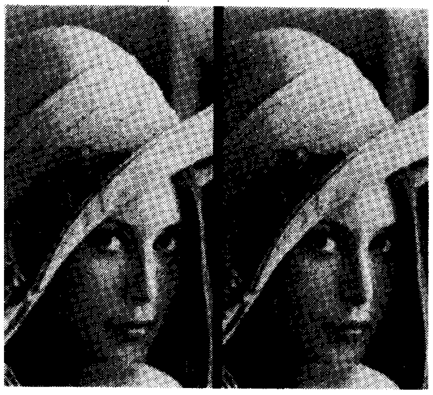


Fig. 8. Coding "Lena": (a) using the GLA codebook, (b) using the SA-C codebook.

TABLE IV
THE PSNR'S FOR THE CODED IMAGES WITH TWO CODEBOOKS. THE DISTORTIONS ARE SUMMED OVER THE ENTIRE IMAGE, AS WELL AS IN A SEGMENTED FASHION WHERE VECTORS HAVING HIGH AND LOW VARIANCES ARE GROUPED TOGETHER

	Codebook Design Method					
	GLA			SA-C		
Image	Total	$\sigma^2 > 144$	$\sigma^2 < 144$	Total	$\sigma^2 > 144$	$\sigma^2 < 144$
Lena	30.48	25.03	34.86	30.59	25.27	34.73
F16-jet	30.33	24.35	37.22	30.74	24.87	37.09
Barb	25.80	22.21	35.33	25.87	22.28	35.18

in Table IV, for the whole image as well as the high and low detail classes. We have included PSNR results for the image "Barb," a 512 × 512 subset of the standard image "Barbara," and "F16-jet" (from the University of Southern California data base) for completeness.

B. Speech and Gauss-Markov Sources

The three algorithms were run on the speech and Gauss-Markov sources using the same type of schedule as in the image source designs, with the value of I in (18) being set to 200. The improvements obtained with each of the these three design algorithms are plotted in Figs. 6 and 7, where it is seen that all perform approximately equal. However, the SR-D and SR-C algorithms again run much faster than the SA-C algorithm, requiring 12 and 100 min of CPU time, respectively, as opposed to 400 min for the SA-C algorithm in a typical run of a 64 codevector speech codebook design. The SA-C algorithm executes more slowly when designing the smaller codebooks because of the increased size of the clusters, and hence the effort needed to compute ΔE, in these cases. Also, since the SA-C algorithm runs a different amount of time at each temperature depending on the size of the training set (which affects L), it will execute faster for the Gauss-Markov designs that use only 1024 training vectors.

It is interesting to note that the SR-D algorithm generally performs slightly better than either the SR-C or the C algorithms, indicating that it may be more effective to perturb the decoder than the encoder. We conjecture that the reason for this difference lies in the amount of state "jittering" done by each perturbation, since each codevector perturbation affects the partition for many training vectors. We expect that exactly equivalent performance will be attained with both of the reduced complexity algorithms if the SR-C case is executed for a sufficiently long time. In theory, if the simplifications to increase computational efficiency are removed, then the SA-C algorithm will eventually produce the global optimum. There is no solid theory to indicate whether the SR-C and SR-D algorithms will reach this same point even given infinite computational resources; however, in practice, the SR-D algorithm seems to be the best choice of quantizer design.

VI. Concluding Remarks

A unified formulation of the vector quantizer design problem using stochastic relaxation has provided a basis for future studies and other design approaches. Several new methods for improving vector quantizer codebook design by combining stochastic relaxation and the GLA algorithm have been analyzed. Other existing techniques are also shown to fit into our general formulation.

We have investigated both simulated annealing procedures as well as reduced complexity techniques that give similar results to SA in much less time. In practice, these new techniques give reliable, reasonably efficient ways to improve the quality of vector quantizers by finding codebooks that are much closer to the global optimum. An important feature of this technique is that the quality of the final codebook achieved, in terms of the MSE when coding the training set, is quite consistent over a wide range of initial conditions for the design. This consistency is important since it removes the initial condition as a parameter in the design (at least with respect to the MSE criterion).

We have demonstrated the effectiveness of the new algorithms by designing codebooks for a wide variety of sources obtained from different applications, including data for image coder design; data for speech coder design; and several different Gauss-Markov sources at different rates. In each case, the SR-GLA algorithms were found to significantly improve the quality of final codebooks, with increased benefit being obtained in those cases where the number of codevectors is large. The reduced complexity algorithms, especially the SR-D, perform as well or better than the direct SA approach and are excellent tools for vector quantizer design.

References

[1] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, pp. 4-29, 1984.
[2] A. Gersho, "On the structure of vector quantizers," *IEEE Trans. Inform. Theory.*, vol. IT-28, no. 2, pp. 157-166, Mar. 1982.

[3] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-95, Jan. 1980.

[4] L. A. Rastrigin, *Random Search in Optimization Problems for Multiparameter Systems* (translated from Russian). Riga, Latvia: "Zinatne" Publishing House, 1965.

[5] B. Widrow and J. M. McCool, "A comparison of adaptive algorithms based on the methods of steepest descent and random search," *IEEE Trans. Antennas Propagat.*, vol. AP-24, pp. 615-637, Sept. 1976.

[6] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-6, pp. 721-741, Nov. 1984.

[7] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Boston: D. Reidel Publishing, 1987.

[8] A. A. El Gamal, L. A. Hemachandra, I. Shperling, and V. K. Wei, "Using simulated annealing to design good codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 116-123, Jan. 1987.

[9] N. Benvenuto and M. Marchesi, "Digital filters design by simulated annealing," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 459-460, Mar. 1989.

[10] T. J. Cornwell, "A novel principle for optimization of the instantaneous Fourier plane coverage of correlation arrays," *IEEE Trans. Antennas Propagat.*, vol. AP-36, pp. 1165-1167, Aug. 1988.

[11] D. J. Goodman and T. J. Moulsley, "Using simulated annealing to design transmission codes for analogue sources," *Electron. Lett.*, vol. 24, no. 10, pp. 617-618, May 1988.

[12] J. R. B. De Marca, N. Farvardin, and Y. Shoham, "Robust vector quantization for noisy channels," in *Proc. Mobile Satellite Conf.*, JPL Pub. 88-9, May 1988, pp. 515-520.

[13] N. Farvardin, "Optimal binary code word assignment for vector quantization over a noisy channel—an application of simulated annealing," presented at the IEEE Int. Symp. Inform. Theory, Kobe, Japan, June 1988.

[14] J. Vaisey and A. Gersho, "Simulated annealing and codebook design," in *Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing*, (New York), Apr. 1988, pp. 1176-1179.

[15] J. K. Flanagan, D. R. Morrell, R. L. Frost, C. J. Read, and B. E. Nelson, "Vector quantization codebook generation using simulated annealing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Glasgow, Scotland), May, 1989, pp. 1759-1762.

[16] A. E. Cetin and V. Weerackody, "Design vector quantizers using simulated annealing," *IEEE Trans. Circuits Syst.*, vol. CAS-35, Dec. 1988.

[17] K. Zeger and A. Gersho, "A stochastic relaxation algorithm for improved vector quantiser design," *Electron. Lett.*, vol. 25, no. 14, pp. 896-898, July 1989.

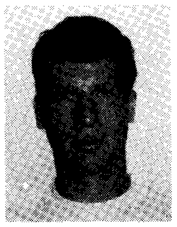
[18] D. P. Connors and P. R. Kumar, "Simulated annealing and balance of recurrence order in time-homogeneous Markov chains," in *Proc. 26th Conf. Decision Contr.*, Dec. 1987, pp. 2261-2263.

[19] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680, May 13, 1983.

[20] N. Metropolis, A. Rosenbluth, M. Rosenbluth, E. Teller, and A. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, pp. 1087-1092, 1953.

[21] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 129-137, Mar. 1982.

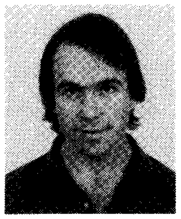
[22] R. M. Gray and E. D. Karnin, "Multiple local optima in vector quantizers," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 2, pp. 256-261, Mar. 1982.



Kenneth Zeger was born in Boston, MA, on August 18,1963. He received both the S.B. and S.M. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (M.I.T.) in 1984, and both the M.A. degree in pure mathematics and the Ph.D. degree in electrical engineering from the University of California, Santa Barbara (UCSB), in 1989 and 1990, respectively.

During 1980-1981 he worked on adaptive antenna arrays for Z-A Inc. in Glenside, PA. He has worked on real-time speech recognition for Hewlett-Packard Company at the General Systems Division in Sunnyvale, CA, and on speech compression techniques at HP Laboratories in Palo Alto, CA, during the years 1982-1985. In 1984 he served as a consultant to Automatic Data Processing Company on digital network design. In July 1990 he joined the electrical engineering faculty at the University of Hawaii as an Assistant Professor. His present research interests include combined source/channel coding, speech and image compression, and computational complexity theory.

Dr. Zeger was awarded a four-year Faculty Development Graduate Fellowship by the American Electronics Association in 1985, was Cochairman of the 1990 IEEE Workshop on Communication Theory, held in Ojai, CA, and received an NSF Presidential Young Investigator Award in 1991.



Jacques Vaisey received the B.Sc. degree from the University of Manitoba, Winnipeg, Canada, in 1980, the M.Sc. degree from Queen's University, Kingston, Ont., in 1982, and the Ph.D. degree from the University of California, Santa Barbara, in 1989, all in electrical engineering.

From February 1982 to September 1984, he was employed in the Transmission Division of Bell-Northern Research (BNR) in Ottawa, Ont., where he worked on the design of 64-QAM and 256-QAM digital radio systems. From September 1989 to September 1990 he worked at INRS-Télécommunications in Montréal as a Research Associate, investigating problems in image coding. He is currently an Assistant Professor in the Department of Engineering Science at Simon Fraser University in Burnaby, British Columbia. His current research interests include image coding, image processing, and multidimensional signal processing.

Allen Gersho (S'58-M'64-SM'78-F'82) received the B.S. degree from the Massachusetts Institute of Technology in 1960 and the Ph.D. degree from Cornell University in 1963.

He is Professor of electrical and computer engineering at the University of California, Santa Barbara (UCSB), and Director of the Center for Information Processing Research at UCSB. He was at Bell Laboratories from 1963 to 1980. His current research activities are in the compression of speech, audio, images, and video signals. He holds patents on speech coding, quantization, adaptive equalization, digital filtering, and modulation and coding for voiceband data modems.

Dr. Gersho served as a member of the Board of Governors of the IEEE Communications Society from 1982 to 1985, and is a member of the Communication Theory Technical Committee and the Signal Processing and Communications Electronics Technical Committee of the IEEE Communications Society. He served as Editor of the IEEE COMMUNICATIONS MAGAZINE and Associate Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS. In 1980, he was awarded the Guillemin-Cauer Prize Paper Award from the Circuits and Systems Society. In 1983, he received the Donald McClennan Meritorious Service Award from the IEEE Communications Society, and in 1984 he was awarded an IEEE Centennial Medal. In 1987 and 1988 he received NASA Tech Brief Awards for technical innovation.