Efficient Bit Allocation for an Arbitrary Set of Quantizers

YAIR SHOHAM, MEMBER, IEEE, AND ALLEN GERSHO, FELLOW, IEEE

Abstract—**This paper proposes a new bit allocation algorithm, capable of efficiently allocating a given quota of bits to an arbitrary set of different quantizers. This algorithm is useful in any coding scheme which employs bit allocation or, more generally, codebook allocation. It produces an optimal or very nearly optimal allocation, while allowing the set of admissible bit allocation values to be constrained to nonnegative integers. It is particularly useful in cases where the quantizer performance versus rate is irregular and changing in time, a situation that cannot be handled by conventional allocation algorithms.**

I. Introduction

BIT allocation is a major concern in any coding scheme where a given quota of bits must be efficiently distributed among a number of different quantizers. In such schemes, each quantizer is capable of operating at (usually, a fixed number of) different rates by using different quantization tables. The rate is usually measured by the minimum number of bits required to address all the entries in the quantization table which is simply $\log_2 N$ where N is the size of the table. The performance of such a quantizer is characterized by its quantizer function (QF), defined by the average quantization distortion as a function of the rate. The set of QF's is used by the allocation algorithm to best determine the allocation strategy that minimizes the overall coding distortion. Most common of such coding schemes are the transform and the subband coders which will frequently be used as examples in various parts of this paper.

The problem of optimal bit allocation was first addressed (within the context of source coding) by Huang and Schultheiss [1] in their original paper on transform coding. They, however, suggested only an approximate solution to the problem. They assumed a set of identical unit variance quantizers each with a simple exponential QF. They solved the allocation problem for this set by allowing the bits to be any arbitrary real numbers, including negative values. Having such a solution at hand, they tried to find, by trial and error, a set of nonnegative integer allocations close to the continuous solution.

An improvement to this method was proposed by Segall [2]. He removed the requirement that the QF's be exponential, but required that they be strictly convex. As before, all the quantizers were assumed identical. The author provided an optimal solution for this case, allowing, again, the allocations to be real numbers which, however, were constrained to be nonnegative.

In most applications, the number of bits that can be assigned to a quantizer is bounded above by some maximum allowable allocation since there is a limit to how fine the quantizer resolution can be. This is particularly important for a vector quantizer since the codebook size grows exponentially with the number of bits. It is sometimes useful to put a lower bound on the number of bits, that is, to specify some minimum nonnegative allowable rate. This reduces the quantization table storage requirements. In subband coding, for example, limiting the minimum rate to 1 or 2 bits is known to produce better perceptual coding quality. The range of admissible rates, specified by the the lower and upper bounds, may, in general, be different for each quantizer. Also, the bit values within the admissible ranges are usually constrained to be integers for practical reasons. Neither of the above-mentioned allocation methods is applicable in the case of different (finite) ranges of admissible integer bit values.

An optimal allocation algorithm for nonnegative integer allocations was proposed by Fox [3]. Unlike Segall's solution, this algorithm (based on so-called marginal return analysis) is applicable to a set of different quantizers. However, the basic assumption is that all the QF's are convex and strictly decreasing with the number of bits. Also, as above, the case of upper and lower bounded allocations is not covered by this algorithm.

In many cases, the quantizers are not identical since the random sequences quantized by them do not have the same statistics, and each quantizer has been optimized for its own input sequence. Consider, for example, the subband coder where different spectral bands of speech are assigned different quantizers. It is known that the statistics of the high-frequency components of speech are significantly different from those of the lower frequency components.

Convexity of the QF's is essential for the above-mentioned algorithm to perform well. However, it may happen that the quantizer functions are not convex. Convexity is suggested by the rate-distortion (RD) theory in cases were the quantizer performance approaches the RD bound, which implies very long block coding. Practical quantizers operate on short data blocks and do not achieve the

Manuscript received September 19, 1986; revised February 15, 1988.
Y. Shoham is with the AT&T Bell Laboratories, Murray Hill, NJ 07974.
A. Gersho is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93016.
IEEE Log Number 8822383.

RD performance, in which case convexity is not guaranteed. Nonconvex QF's may be generated by a set of quantization tables where not all of them perfectly match the source statistics. This usually happens when a quantization table (more generally, a vector codebook) is built of a multiple of subtables. A typical example to this case is the multistage vector quantizer [9] which is known to have an irregular QF. Nonconvexity of the QF may also happen if the set of quantization tables are not all equally well designed, or if the changing characteristics of a nonstationary source renders some of the tables inefficient. In extreme cases, the QF's may not even be monotone-nonincreasing with the number of bits. This strange anomaly may arise, for example, in multistage quantizers if adding one more bit to the rate by using one more 1-bit codebook is very inefficient and actually increases the distortion. The allocation algorithms mentioned above are not suitable for nonconvex or nonmonotone decreasing QF.

An optimal solution to the allocation problem can be obtained via dynamic programming [4]–[6]. However, the complexity of this technique is rather high. If R bits are to be allocated to a set of quantizers, dynamic programming requires on the order of R^2 arithmetic operations per quantizer. In coding schemes which use long blocks of data, the number of bits per block (R) is usually very high (hundreds, if not thousands), which makes dynamic programming unacceptable for this application.

In this paper we propose a new allocation algorithm, which efficiently solves the allocation problem for an arbitrary set of quantizers, under all the constraints mentioned above, with an acceptable degree of complexity. This algorithm can be applied to any coding scheme where bit allocation is performed. Moreover, since the algorithm performs efficiently for any set of quantizers, the allocation procedure can be dynamically updated by an on-line measurement of the distortion versus rate functions of all the quantizers. In particular, one can perform the "ultimate bit allocation" by measuring the distortion function for each single block. In other words, quantize each entry of the current block by its own quantizer, with all the admissible bit values, record the resulting distortions, and form the QF's. Then, apply the allocation algorithm and get the final bit allocation. This is, of course, a tedious operation, but may very well be worth it in terms of the increase in performance. Note that since, in this case, the measured distortions are not averaged, the QF's may be very erratic, that is, not convex and not even always decreasing with the rate. The main point is that such an ultimate bit allocation is fundamentally impossible with the traditional allocation algorithms.

II. The Allocation Problem

The allocation problem can be formulated as follows. We are given a set of M variable-rate quantizers and a set of M gains represented by the vector $G = (g_1, g_2, \cdots, g_M)$. The input to the kth quantizer is first normalized by the corresponding gain g_k and then quantized with i bits where i belongs to the set of nonnegative integer values $\{p_k, \cdots, q_k\}$. The resulting squared errors for all admissible bit values form the *normalized quantizer function* (NQF) of the kth quantizer, denoted by $\{\Theta_k(i), i = p_k, p_k + 1, \cdots, q_k\}$. Thus, if i bits are used, the actual squared error at the kth quantizer output is $(g_k)^2 \Theta_k(i)$.

Given a quota of bits R, the problem is to find the best way of distributing these bits among the quantizer so as to minimize the total squared error. In other words, find an allocation vector $B = (b_1, b_2, \cdots, b_M)$ which minimizes

$$\sum_{k=1}^{M} (g_k)^2 \Theta_k(b_k)$$

subject to

$$p_k \le b_k \le q_k, \qquad k = 1, \cdots, M$$

and

$$\sum_{k=1}^{M} b_k \le R.$$

Note that the domains $\{p_k, \cdots, q_k\}$ of admissible bit assignments are different for each quantizer (each k). In this work, we impose a practical constraint of nonnegative integer bit values. However, the algorithm can be generalized to allow for any finite set of real numbers as admissible bit assignments. Also, we obviously have $p_k \le q_k \le R$ for all k.

A typical example where such an allocation problem arises is subband coding. In this coding system, signals of several spectral bands are segmented into fixed-length blocks and quantized each by its own separate quantizer. The square root of the block energy is defined as the block gain. The data in the block are first normalized by the block gain and then fed to the quantizer. The (scalar or vector) quantizer is tuned to the characteristics of the normalized data and is able to operate at a predetermined range of bit rates. The problem, precisely formulated above, is how to split a total of R bits (the overall system rate) among all the bands so as to minimize the overall distortion, based on the measured gains and the quantizer characteristics.

The algorithmic solution to the problem, proposed here, is based on a little-known version of the Lagrange-multiplier method, as applied to allocation problems. In the next section, we state the main theorem associated with this method, which is the basis for our algorithm. Following this, the algorithm is developed and its efficiency is demonstrated and discussed.

III. Introduction of the Main Theorem

Let S be the finite set of all admissible allocation vectors. Let $H(B)$ be some real-valued function called the objective-function of B, defined for all allocation vectors B in S. Let $R(B)$ be some real-valued function called the constraint-function of B, defined for all B in S. Consider the following problem (called the constrained problem).

Given a constraint R_c, find

$$\min_{B \in S} H(B) \tag{1a}$$

subject to

$$R(B) \leq R_c. \tag{1b}$$

Theorem: For any $\lambda \geq 0$, the solution $B^*(\lambda)$ to the unconstrained problem

$$\min_{B \in S} \{H(B) + \lambda R(B)\} \tag{2}$$

is also the solution to the constrained problem (1) with the constraint $R_c = R(B^*(\lambda))$, that is, with $R(B) \leq R(B^*(\lambda))$. To simplify notation, we define $R^*(\lambda) = R(B^*(\lambda))$.

The proof of this theorem (which is actually quite simple) and some of its applications can be found in [7]. For completeness we repeat the proof here.

Proof: For the solution B^*, we have

$$H(B^*) + \lambda R(B^*) \leq H(B) + \lambda R(B)$$

for all B in S. Equivalently, we have

$$H(B^*) - H(B) \leq \lambda(R(B) - R(B^*))$$

for all B in S. Since this is true for all B in S, it is obviously true for all B in any subset of S. In particular, it is true for all B in a subset defined by

$$S^* = \{B: R(B) \leq R(B^*)\}.$$

Since $\lambda \geq 0$ we have

$$H(B^*) - H(B) \leq 0, \qquad B \in S^*$$

that is, B^* is the solution to the constrained problem with $R_c = R(B^*)$, and the theorem is proved.

It is important to properly appreciate what this theorem says. It does not guarantee any solution to the constrained problem (1). All it says is that to every nonnegative λ, there is a corresponding constrained problem whose solution is identical to that of the unconstrained problem. The main point is, however, that if $R^*(\lambda)$ happens to be equal to R_c, then $B^*(\lambda)$ is the desired solution to the constrained problem.

As we sweep λ over the range zero to infinity, sets of solutions $B^*(\lambda)$ and constraints $R^*(\lambda)$ are created. The key questions are whether or not one value of $R^*(\lambda)$ coincides with R_c and how to find the corresponding λ. These questions are discussed throughout the development of the algorithm which is based on the above theorem.

IV. Development of the Algorithm

The bit allocation problem formulated in the previous section is a constrained problem specified by

$$S = \{B: b_k \in \{p_k, \cdots, q_k\}, \quad k = 1, \cdots, M\} \tag{3}$$

$$H(B) = \sum_{k=1}^{M} (g_k)^2 \Theta_k(b_k) \tag{4}$$

$$R(B) = \sum_{k=1}^{M} b_k. \tag{5}$$

For the purpose of this discussion, the squared gains and the normalized quantizer functions can be represented by

$$W_k(b_k) = (g_k)^2 \Theta_k(b_k), \qquad k = 1, \cdots, M. \tag{6}$$

Thus, the unconstrained problem can be written as

$$\min_{B \in S} \left\{ \sum_{k=1}^{M} W_k(b_k) + \lambda \sum_{k=1}^{M} b_k \right\}. \tag{7}$$

The crucial point is that the solution is obtained by minimizing each term of the sum in (7) separately. Denote by $b_k^*(\lambda)$ the kth component of $B^*(\lambda)$ and define

$$S_k = \{p_k, \cdots, q_k\} \tag{8}$$

then $b_k^*(\lambda)$ solves

$$\min_{i \in S_k} \{W_k(i) + \lambda i\}. \tag{9}$$

Given λ, one can solve for all b_k^*'s, sum them all up to get $R^*(\lambda)$, and then compare this value to the desired constraint R_c. If $R^*(\lambda) = R_c$, the desired solution has been achieved. $R^*(\lambda)$ is called the *solution rate* since it is the total number of bits associated with the solution $B^*(\lambda)$.

The solution $b_k^*(\lambda)$ may not be unique. The set of solutions to (9) for the kth quantizer and for a given λ are denoted by $\{b_k^*(\lambda)\}$. Therefore, the solution $B^*(\lambda)$ is also not necessarily unique. We denote the set of solutions for a given λ by $\{B^*(\lambda)\}$. The corresponding set of solution rates is denoted by $\{R^*(\lambda)\}$. The size of the solution set $\{B^*(\lambda)\}$ is at least the size of the set $\{R^*(\lambda)\}$, but may be larger since more than one solution may yield the same solution rate.

Let G be the (finite) set of all solution rates $\{R^*(\lambda)\}$ obtained by sweeping λ from 0 to ∞. A vector B, for which $R(B)$ is not in G, is called an inaccessible point in S since there is no λ for which $B = B^*(\lambda)$. It is desirable that the solution to our constrained problem not be an inaccessible point since an inaccessible solution cannot be found by solving the unconstrained problem for λ.

To proceed further in the development of the algorithm, the following lemmas are needed.

Lemma 1: Let $W(b)$ be a real-valued function over some bounded and closed domain Z on the real line. Let b_1 be a solution to

$$\min_{b \in Z} \{W(b) + \lambda_1 b\}$$

and let b_2 be a solution to

$$\min_{b \in Z} \{W(b) + \lambda_2 b\}$$

then,

$$(\lambda_1 - \lambda_2)(b_1 - b_2) \leq 0$$

for any function $W(b)$.

In other words, as λ increases, the minimizing value of b either decreases or remains unchanged.

Proof: By definition of b_1 and b_2, we have

$$W(b_1) + \lambda_1 b_1 \leq W(b_2) + \lambda_1 b_2$$
$$W(b_2) + \lambda_2 b_2 \leq W(b_1) + \lambda_2 b_1.$$

From this we get

$$W(b_1) - W(b_2) \leq \lambda_1(b_2 - b_1)$$
$$W(b_2) - W(b_1) \leq \lambda_2(b_1 - b_2).$$

Summing both sides of these two inequalities we get

$$0 \leq (b_1 - b_2)(\lambda_2 - \lambda_1)$$

and the lemma is proved.

Lemma 2: The solutions $b_k^*(\lambda)$ for all k and the corresponding solution rates $R^*(\lambda)$ are monotonically nonincreasing with λ, that is, if $\lambda_2 \geq \lambda_1 > 0$, then $b_k^*(\lambda_2) \leq b_k^*(\lambda_1)$ and $R^*(\lambda_2) \leq R^*(\lambda_1)$.

Proof: Apply Lemma 1 to (9) which proves Lemma 2 for $b_k^*(\lambda)$ for all k, and hence for the sum $R^*(\lambda)$.

Lemma 3: The solution sets $\{B^*(\lambda_1)\}$ and $\{B^*(\lambda_2)\}$ for two different values λ_1 and λ_2 can have at most one solution in common.

Proof: Suppose $\lambda_2 > \lambda_1$. By Lemma 2, the lowest value in $\{b_k^*(\lambda_1)\}$ is greater or equal to the greatest value in $\{b_k^*(\lambda_2)\}$. Therefore, at most one value (the lowest for λ_1 which could be equal to the greatest for λ_2) can be common to both sets. Since this is true for all k, the lemma is proved.

A corollary of this lemma is that two distinct values of λ cannot have the same set of solutions, unless this set contains a single solution. This implies that a set of multiple solutions $\{B^*(\lambda)\}$ occurs at an isolated value of λ since $\lambda + \epsilon$ has a different set of solutions $\{B^*(\lambda + \epsilon)\}$ (with at most one solution in common), regardless of how small $|\epsilon|$ is, as long as $\epsilon \neq 0$. The values of λ which correspond to multiple solutions are called *singular values* of λ.

Lemma 4: Let λ have a single solution $B^*(\lambda)$. If singular values less than λ exist, then the nearest one to λ, denoted by λ_1, has a solution $B^*(\lambda_1) = B^*(\lambda)$. Similarly, if singular values greater than λ exist, then the nearest one to λ, denoted by λ_2, has a solution $B^*(\lambda_2) = B^*(\lambda)$.

Proof: Since λ is nonsingular we have from (7)

$$W_k(b_k^*) - W_k(b_k) < \lambda(b_k - b_k^*); \quad b_k \neq b_k^*;$$
$$k = 1, \cdots, M.$$

The multiple solutions associated with λ_2 imply that there are k's for which there are b_k's such that $b_k - b_k^* < 0$. Therefore, there exists $\epsilon > 0$ such that

$$W_k(b_k^*) - W_k(b_k) < (\lambda + \epsilon)(b_k - b_k^*)$$
$$k: b_k - b_k^* > 0$$

$$W_k(b_k^*) - W_k(b_k) = (\lambda + \epsilon)(b_k - b_k^*)$$
$$k: b_k - b_k^* < 0.$$

Let ϵ_m^+ be the minimum value of an ϵ over all k such that $b_k - b_k^* < 0$. Clearly, $\lambda + \epsilon_m^+$ is singular. It has at least two solutions, one of which is $B^*(\lambda)$. Also, $\lambda + \epsilon_m^+ = \lambda_2$ since it is the closest singular value to λ.

Using the same arguments, it is easily shown that there exists $\epsilon_m^- < 0$ such that $\lambda + \epsilon_m^- = \lambda_1$ and λ_1 has the solution $B^*(\lambda)$.

This lemma has two important corollaries. First, two adjacent singular values have one and only one solution in common. Second, all nonsingular values of λ between two adjacent singular values have the same (single) solution.

Fig. 1 illustrates a typical dependency of the solution rate $R^*(\lambda)$ on λ. It shows a decreasing staircase curve where the discontinuities correspond to singular values of λ. The black dots at the singular points represent multiple values of constraints which correspond to multiple solutions.

The main conclusion from the above lemmas is that it is not necessary to sweep λ over all possible values (all values on the nonnegative real line) in order to find all the solutions to the unconstrained problem. It is sufficient to check only the singular points, that is, only those values of λ for which more then one solution exists.

To find a constraint which satisfies $R^*(\lambda) = R_c$ (if it exists), it is not necessary to search over all the singular values. Because of the monotonicity of $R^*(\lambda)$, only a small subset of singular λ's needs to be searched. The search strategy which is the main issue in the proposed algorithm will soon be discussed.

Locating the desired solution requires the knowledge of the singular λ's. For this we have Lemma 5.

Lemma 5: Let λ_1 be singular, $\lambda > \lambda_1$, and $B^*(\lambda)$ be a common solution associated with λ and λ_1. Define the set

$$S_k^+ = \{b_k^* + 1, \cdots, q_k\}. \tag{10a}$$

This set is nonempty if $b_k^* + 1 \leq q_k$. Also, define the set

$$M^+ = \{k \in \{1, \cdots, M\} : S_k^+ \text{ is nonempty}\}. \tag{10b}$$

This defines the set of all the QF's for which an increase in bit assignment is possible. Then, the singular value λ_1, which is necessarily the closest to λ from below, is given in terms of λ by

$$\lambda_1 = \max_{k \in M^+} \max_{i \in S_k^+} \frac{W_k(b_k^*(\lambda)) - W_k(i)}{i - b_k^*(\lambda)}. \tag{10c}$$

Proof: Let $B^*(\lambda_1)$ be the common solution, that is,

$$b_k^*(\lambda_1) = b_k^*(\lambda), \qquad k = 1, \cdots, M.$$

Since λ_1 is singular, it has at least one more solution in addition to $B^*(\lambda_1)$. This means that there exists at least

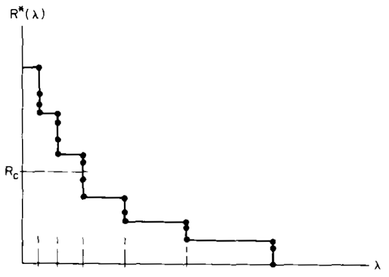


Fig. 1. Typical dependency of the solution rate $R^*(\lambda)$ on the Lagrange multiplier λ.

one value of k and at least one value of i such that

$$k \in M^+$$
$$i \in S_k^+$$

and

$$W_k(b_k^*(\lambda_1)) + \lambda_1 b_k^*(\lambda_1) = W_k(i) + \lambda_1 i.$$

For other values of k and i, we have

$$W_k(b_k^*(\lambda_1)) + \lambda_1 b_k^*(\lambda_1) \le W_k(i) + \lambda_1 i.$$

Now, replace $b_k^*(\lambda_1)$ by $b_k^*(\lambda)$ in the last two expressions, and express λ_1 as a function of the other terms. This gives

$$\lambda_1 \ge \frac{W_k(b_k^*(\lambda)) - W_k(i)}{i - b_k^*(\lambda)}, \qquad i \in S_k, \qquad k \in M^+.$$

The equality in the last expression holds if and only if i is a solution for some k in M^+. To find λ_1, we search for the maximum of the right term in the last inequality, that is, compute (10c). Thus, Lemma 5 is proved.

Note that if the sets S_k^+ are empty for all k, then there is no λ_1 satisfying $\lambda_1 < \lambda$ which means that there are no singular points below λ.

The pair k', i' which solves (10c) determines a solution to the unconstrained problem for λ_1. This solution is obtained by setting $b_{k'}^*(\lambda_1) = i'$ and $b_k^*(\lambda_1) = b_k^*(\lambda)$ for all other values of k.

Similarly, we have Lemma 6.

Lemma 6: Let λ_2 be singular, $\lambda_2 > \lambda$, and $B^*(\lambda)$ a common solution associated with λ and λ_2. Define the set

$$S_k^- = \{p_k, \cdots, b_k^*(\lambda) - 1\} \tag{11a}$$

which is nonempty only if $b_k^*(\lambda)) \ge 1$. Define the set

$$M^- = \{k : k \in \{1, \cdots, M\}; \quad S_k^- \text{ is nonempty}\}. \tag{11b}$$

Then, the singular value λ_2 is given in terms of λ by

$$\lambda_2 = \min_{k \in M^-} \min_{i \in S_k^-} \frac{W_k(i) - W_k(b_k^*(\lambda))}{b_k^*(\lambda) - i} \tag{11c}$$

which can be proved in the same way as for (10c). Note that if S_k^- are empty for all k, there is no λ_2 which satisfies $\lambda_2 > \lambda$.

As for (10c), the pair k', i' which solves (11c) determines a solution to the unconstrained problem for λ_2. This solution is obtained by setting $b_{k'}^*(\lambda_2) = i'$ and $b_k^*(\lambda_2) = b_k^*(\lambda)$ for all other values of k.

The last two lemmas say that by knowing a singular value, the next from below or from above can be found, and thus, all singular values and all the solutions $B^*(\lambda)$ can be located. This motivates the following basic search procedure which is the essence of the algorithm.

1) Obtain an initial value λ (in a way soon to be discussed).

2) Solve the unconstrained problem by using (9). If λ is not singular, there is only one such solution and one constraint $R^*(\lambda)$. If λ is singular, then there are at least two different solutions. Find the two solutions from $\{B^*(\lambda)\}$ with greatest and smallest constraints denoted by $R_l^*(\lambda)$ and $R_h^*(\lambda)$, respectively.

3) If the desired constraint R_c is such that

$$R_l^*(\lambda) \le R_c \le R_h^*(\lambda),$$

then obtain all solutions in $\{B^*(\lambda)\}$ [using (9)] and find the one for which the constraint, denoted by $R_a^*(\lambda)$, is the closest to R_c from below. If $R_c = R_a^*(\lambda)$, an optimal solution has been found. If not, an approximate solution, denoted by B_a^*, has been found. Stop the search.

4) If $R_c < R^*(\lambda)$ [or $R_c < R_l^*(\lambda)$], find the next singular λ using (11) [and the current solution $B^*(\lambda)$] which corresponds to $R^*(\lambda)$ [or $R_l^*(\lambda)$]. Then, go to Step 2.

5) If $R_c > R^*(\lambda)$ [or $R_c > R_h^*(\lambda)$], find the next singular λ using (10) and the current solution which corresponds to $R^*(\lambda)$ [or $R_h^*(\lambda)$]. Then, go to Step 2.

If the search terminates with only an approximate solution, which means that the optimal solution is an inaccessible point, then this solution can be adjusted by adding a few bits until the sum of the modified allocations becomes equal to R_c. The criterion for adding bits is to cause the greatest reduction in the distortion $H(B^*(\lambda))$. We add one bit at a time according to the following steps.

6) Find k' which solves

$$\min_{k \in M^+} \{W_k(b_k^* + 1) - W_k(b_k^*)\}. \tag{12}$$

7) Update the solution by adding 1 to $b_{k'}^*$.

8) Repeat Steps 6 and 7 until the sum of the allocations equals R_c.

Since B_a^* is not an optimal solution, it is natural to ask how far it is from the desired optimal solution. With the aid of the following lemma, a bound on the difference between the distortions of the optimal and approximate solutions can be found.

Lemma 7: Let B be the desired solution of the original constrained problem with the constraint R_c. Let $B^*(\lambda_1)$ and $B^*(\lambda_2)$ be two solutions such that

$$R^*(\lambda_1) \le R_c \le R^*(\lambda_2). \tag{13a}$$

Then,

$$H(B^*(\lambda_2)) \le H(B) \le H(B^*(\lambda_1)). \tag{13b}$$

Proof: The left side of (13a) directly implies the right side of (13b) by the main theorem. For the left side of (13b) we have

$$H(B^*(\lambda_2)) + \lambda_2 R^*(\lambda_2) \le H(B) + \lambda_2 R(B) \le H(B) + \lambda_2 R_c.$$

Since $\lambda_2 \ge 0$ we have

$$H(B^*(\lambda_2)) - H(B) \le \lambda_2(R_c - R^*(\lambda_2)) \le 0$$

which means that the right side of (13b) implies the left side of (13b) and the lemma is proved.

Now, let $B^*(\lambda_1)$ be the solution at the end of the search (Step 3). By adding more bits to this solution, we reduce the distortion. However, upon reaching $R(B_a^*) = R_c$, the distortion $H(B_a^*)$ cannot be lower than $H(B)$ and cannot be higher than $B^*(\lambda_1)$. This means that (13b) still holds if we replace $H(B)$ by $H(B_a^*)$, which implies the following:

$$H(B_a^*) - H(B) \le H(B^*(\lambda_1)) - H(B^*(\lambda_2)). \quad (14a)$$

Thus, the difference in the distortions associated with the optimal and approximate solutions to the constrained problem is within a range determined by two solutions to the unconstrained problem which are the closest to the desired solution from above and below.

A more useful bound is expressed terms of the difference in decibels between the distortions of the optimal and approximate solutions. The Allocation Performance Bound (APB) is defined by

$$APB = -10 \log_{10} \frac{H(B_a^*) - H(B^*(\lambda_1)) + H(B^*(\lambda_2))}{H(B_a^*)} \quad (14b)$$

and from (14a) we have

$$10 \log \frac{H(B_a^*)}{H(B)} \le APB. \quad (14c)$$

This bound can be computed in an actual application of the algorithm to evaluate the approximation.

V. Variations of the Basic Algorithm

The computational effort associated with the basic algorithm described in the previous section can be significantly reduced, based on the following observation.

Let λ_1, λ_2, and λ_3 be three Lagrangian multipliers which satisfy

$$\lambda_1 \le \lambda_2 \le \lambda_3. \quad (15)$$

Then, by virtue of Lemma 2, the corresponding solutions satisfy

$$b_k^*(\lambda_3) \le b_k^*(\lambda_2) \le b_k^*(\lambda_1), \qquad k = 1, \cdots, M. \quad (16)$$

This means that, given $B^*(\lambda_1)$ and $B^*(\lambda_3)$, the solution for λ_2 can be searched for in a considerably smaller range than that specified in (9)–(11). The search regions for these expressions [see (8), (10a), and (11a)] are now redefined as

$$S_k = \{i : b_k^*(\lambda_3) \le i \le b_k^*(\lambda_1)\}, \quad k = 1, \cdots, M \quad (17)$$

$$S_k^+ = \{b_k^*(\lambda_3) + 1, \cdots, b_k^*(\lambda_1)\}, \quad k = 1, \cdots, M \quad (18)$$

$$S_k^- = \{b_k^*(\lambda_1), \cdots, b_k^*(\lambda_3) - 1\}. \quad (19)$$

With these modified search regions, we can describe the first variant of the basic algorithm.

Variant 1:

1) Find initial values λ_1 and λ_3 which satisfy

$$R^*(\lambda_1) \le R_c \le R^*(\lambda_3). \quad (20)$$

One way to do this is as follows.

1.1) Find an initial λ_1 (to be discussed in the next section).

1.2) Solve (9) with the old S_k as in (8) for all k and get $R^*(\lambda_1)$.

1.3) If $R^*(\lambda_1)$ is greater than R_c, set $\lambda_3 = \lambda_1$ and $B^*(\lambda_3) = B^*(\lambda_1)$. Increase λ_1 by adding to it some preselected update constant. Go to Step 1.2.

1.4) If $R^*(\lambda_1)$ is lower than R_c, decrease λ_1 by subtracting the update constant. Go to Step 1.2.

1.5) Repeat the above until the desired λ_1 and λ_3, for which (20) holds, have been obtained.

2) Continue as in the basic procedure but with the modified search regions as in (17)–(19). Note that as new solutions for λ's are found, the search regions can be dynamically modified, thereby, continuously decreasing the search intensity.

The value of the update constant of Step 1 determines the rate at which the procedure approaches the desired two initial solutions. If this constant is large, the two solutions can be obtained after one or two iterations. However, they may be far apart which means large search regions. If this constant is very small, many iterations are needed, but the resulting search regions are small since the two solutions will be close to each other. The best update constant should be experimentally found by actually measuring the overall search complexity while running the algorithm with various values of this constant.

It was observed during an experimental evaluation of the algorithm that more than two solutions for singular λ's were never detected. Therefore, we may assume that there cannot be more than two solutions for any singular λ with a very small risk, if at all, of skipping a solution which might have been the desired one.

Under this assumption, it is not necessary to search for all the solutions of a singular λ. The solution found by computing either (10c) or (11c) is the new solution of the new λ. This means that once λ_1 and λ_2 have been found, (9) need not be solved in order to find the next λ. This considerably reduces the search intensity.

We therefore have a new variant of the algorithm described as follows.

Variant 2:

1) Find two initial values λ_1 and λ_3 as in Variant 1.

2) Start from λ_3 and obtain successive singular values and solutions by successively solving (10c) for new λ_3 and new solution $B^*(\lambda_3)$. Update the search regions as described in Variant 1.

3) When $R^*(\lambda_3)$ becomes less than or equal to R_c, stop the search.

4) If necessary, adjust the solution $B^*(\lambda_3)$ to obtain a final approximate solution B_a^* as in Steps 6–8 of the basic algorithm.

VI. Initial Lagrange Multiplier

The computational complexity of the algorithm strongly depends on the initial λ since the closer this λ is to the final value, the fewer iterations are needed. The following method of obtaining an initial λ was empirically found to yield a low average computational intensity. The method is based on the fact that a necessary condition for a set $\{b_k\}$ to be a solution is

$$W_k(b_k) - W_k(b_k + 1) \leq \lambda \leq W_k(b_k - 1) - W_k(b_k)$$
$$p_k + 1 \leq b_k \leq q_k - 1$$
$$k = 1, 2, \cdots, M$$

which can very easily be verified. Since the above is true for all k, we can get an "average" initial λ by summing either one of the two inequalities in the last expression with respect to k. Using the inequality on the left, we obtain

$$\lambda \approx \frac{1}{M} \sum_{k=1}^{M} \{W_k(b_k) - W_k(b_k + 1)\}.$$

The b_k's are obviously unknown, therefore, we replace them by an "average" of their admissible values. The value $a_k = \text{INT}\{(q_k + p_k)/2\}$, where INT $\{\cdot\}$ means rounding to the nearest integer, was experimentally found to be a good replacement to the b_k's for this purpose. The initial λ is, therefore, computed by

$$\lambda = \frac{1}{M} \sum_{k=1}^{M} \{W_k(a_k) - W_k(a_k + 1)\}. \quad (21)$$

However, this initialization method is not necessarily the best one, and further study of this problem may yield a better solution.

VII. Evaluation of the Algorithm

In this section, we provide experimental results which demonstrate the efficiency of the proposed allocation algorithm. The algorithm was applied to a sequence of 1000 gain vectors $G_n = (g_{1,n}, \cdots, g_{64,n})$, $n = 1, \cdots, 1000$, with a given set of 64 NQF's $\Theta_k(b)$, $k = 1, \cdots, 64$. Three different sets of NQF's were considered: a set of 64 identical exponential NQF's over the bit range [0, 24] (Set 1); a set of 64 identical but nonconvex NQF's over [0, 24] (Set 2), and a set of different nonconvex NQF's each defined over a specific range $[0, q_k]$, $k = 1, \cdots, 64$ (Set 3).

The gain vectors were computed from a sequence of 1000 DCT blocks of length 256, extracted from a speech waveform. Each DCT block was partitioned into 64 4-dimensional subvectors and the RMS energy of the ith subvector was defined as g_i.

The experiment with Set 1 simulated a situation where all 64 4-D DCT subvectors had the same statistical characteristics and they were all quantized by the same *ideal* 4-D vector quantizer whose NQF was assumed to be $\Theta_k(i) = 2.2 \times 10^{0.5i}$. The quantizer factor 2.2 was actually obtained by averaging the performance of a 4-D vector quantizer over a long training set of 4-D normalized DCT subvectors.

The experiment with Set 2 simulated the case where all the subvectors in the DCT block were quantized by the same three-stage vector quantizer. Details on the structure of such a quantizer can be found in [8]. The main point is that the NQF of this quantizer is not convex, that is, the difference function $\Theta_k(i + 1) - \Theta_k(i)$ is not monotonically decreasing. This behavior is caused by the transitions from one stage to another in the quantization process. The NQF used in this experiment is depicted in Fig. 2. The figure shows the normalized distortion in decibels (to capture the whole range) versus number of bits.

The experiment with Set 3 simulated the case where each subvector in the DCT block was quantized by a separate vector quantizer, tuned to statistics of that subvector. Again, three-stage vector quantization was assumed. Furthermore, each quantizer had a different upper bound on the admissible bit assignment. The lowest bit assignment was zero for all the quantizers. The set of the 64 NQF's, which were found by actually designing 64 three-stage VQ's over a long training set of 4-D DCT subvectors, is shown in Fig. 3. The NQF's are shown in 6 separate groups of graphs for better clarity. A straight horizontal line at the end of a graph indicates a region of inadmissible bit assignments. In other words, an NQF is defined up to the beginning of such a line. Observe that these NQF's are not only nonc nvex, they are sometimes not even monotonically decrea ng.

The allocation algorithm was applied to the sequence of 1000 gain vectors with Set 1, Set 2, and Set 3. The total number of bits to allocate to each block was 128, corresponding to 2 bits per gain (0.5 bit per subvector component). In each case we measured the following.

1) The percentage of truly optimal allocations, out of the total 1000 allocations.

2) The signal-to-distortion ratio (SNR), defined as the ratio of the mean square average of the gain sequence to the average distortion, where the distortion was calculated for each block according to (4).

3) The average and maximum bounds (APB) on the deviation in performance from that of the optimal solution [see (14)].

4) The average of the arithmetic operations (mul. add.

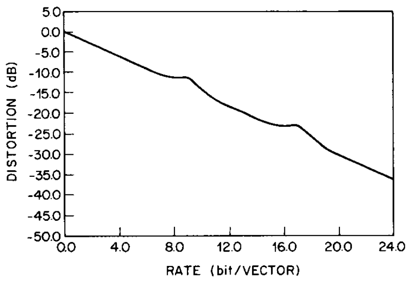


Fig. 2. Quantizer function of Set 2.

TABLE I
EXPERIMENTAL RESULTS

QUANTIZER SET	OPTIMAL ALLOCATIONS (%)	SNR (dB)	AVERAGE APB (dB)	MAXIMUM APB (dB)	AVERAGE OP./GAIN
SET 1	100	10.9	0	0	212
SET 2	82	14.3 (FOX: 10.6)	0.21	0.36	204 (FOX: 275)
SET 3	58	12.4	0.17	0.53	191

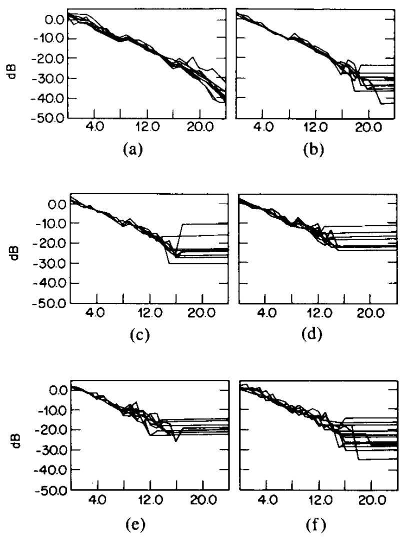


Fig. 3. The 64 quantizer functions of Set 3. (a) $k = 1, \cdots, 10$. (b) $k = 11, \cdots, 20$. (c) $k = 21, \cdots, 30$. (d) $k = 31, \cdots, 40$. (e) $k = 41, \cdots, 50$. (f) $k = 51, \cdots, 64$.

compare) performed by the algorithm per gain component.

The experimental results are summarized in Table I. As shown, the algorithm was fully optimal (100 percent optimal allocations) in the case of Set 1. It is not difficult to prove that this is always the case for convex NQF's. It simply means that in this case, there are no inaccessible points in the allocation space.

In the case of Set 2, 82 percent of the allocations were optimal. The status of the rest was unknown. However, for the 18 percent possibly nonoptimal allocations, the average bound on the increase in the distortion (compared to that of an optimal allocation) was only 0.21 dB and the maximum bound was 0.36 dB. This shows that although it is not strictly optimal in all cases, the algorithm is highly efficient.

We also applied the allocation algorithm proposed by Fox [3] to Set 2 and compared the resulting SNR to our result. As shown, the SNR produced by Fox's algorithm was 3.7 dB lower. This means that Fox's algorithm, which is known to be optimal for convex NQF's, breaks down when applied to a set of nonconvex NQF's, and may not even be used as an approximation to the optimal algorithm.

For Set 3, 58 percent of the allocations were optimal. The rest were unknown. The average bound, however, was 0.17 dB and the maximum one was 0.53 dB. This demonstrates that even in the case of nonconvex, nondecreasing, and nonidentical QF's, the algorithm still is essentially optimal.

VIII. CONCLUSION

In this paper, we propose an efficient bit allocation algorithm for an arbitrary set of quantizers. In contrast to earlier bit allocation algorithms, this algorithm is not based on prior assumptions about the nature of the given set of quantizers. Rather, bit allocation is optimized for the true performance of each individual quantizer, as measured from the data. The algorithm can be used in any coding scheme which requires bit allocation or, more generally, codebook allocation.

The algorithm is based on the Lagrangian method for integer allocation. Two variants of the basic algorithm are described. More work can be done to improve the algorithm and reduce its complexity, based on the same basic idea.

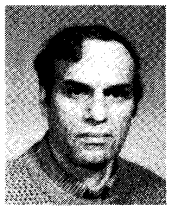
Experimental results, provided in this paper, demonstrate that the algorithm is essentially optimal and can be very useful in many applications.

REFERENCES

[1] J. J. Huang and P. M. Schultheiss, "Block quantization of correlated Gaussian random variables," *IEEE Trans. Commun. Sys.*, vol. CS-11, pp. 289-296, Sept. 1963.
[2] A. Segall, "Bit allocation and encoding for vector sources," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 162-169, Mar. 1976.
[3] B. Fox, "Discrete optimization via marginal analysis," *Manage. Sci.*, vol. 13, no. 3, pp. 210-216, Nov. 1966.
[4] O. A. Zuniga, R. M. Haralick, and R. L. Klein, "Adaptive image data compression," in *Proc. IEEE ICASSP-79*, Feb. 1979, pp. 583-587.
[5] A. V. Trushkin, "Bit number distribution upon quantization of a multivariate random variable," *Problems of Inform. Transmission*, vol. 16, pp. 76-79, 1980 (Translated from Russian).
[6] ——, "Optimal bit allocation algorithm for quantizing a random vector," *Problems of Inform. Transmission*, vol. 17, pp. 156-161, 1981 (Translated from Russian).
[7] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operation Res.*, vol. 11, pp. 399-417, 1963.

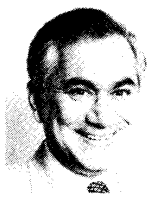
[8] Y. Shoham, "Hierarchical vector quantization with application to speech waveform coding," Ph.D. dissertation, Univ. Calif. at Santa Barbara, Dep. Elec. Comput. Eng., Mar. 1985.

[9] B. H. Juang and A. H. Gray, "Multiple stage vector quantization for speech coding," in *Proc. Int. Conf. ASSP*, Paris, France, May 1982, pp. 597-600.



Yair Shoham (S'82-M'85) received the B.Sc. and M.Sc. degrees from the Technion-Israel Institute of Technology, Haifa, Israel, in 1969 and 1973, respectively, and the Ph.D. degree in 1985 from the University of California at Santa Barbara.

In 1969 he joined RAFAEL, the Armament Development Authority, Israel, where he was involved in the development of special-purpose tactical communication circuits and systems. From 1978 to 1980 he was a Group Head, responsible for various projects in the area of point-to-point communication links. In 1985-1986 he was a Consultant for AT&T Bell Laboratories, and in 1986 he joined the Signal Processing Research Department of Bell Laboratories. His current interests are in the areas of digital signal processing and vector quantization algorithms, with applications to digital speech coding.



Allen Gersho (S'58-M'64-SM'78-F'82) received the B.S. degree from M.I.T., Cambridge, in 1960 and the Ph.D. degree from Cornell University, Ithaca, NY, in 1963.

He was at Bell Laboratories from 1963 to 1980. He is now a Professor of Electrical and Computer Engineering at the University of California, Santa Barbara. His current research activities are in speech and image compression using vector quantization techniques, and in pattern recognition and neural networks. He holds patents on adaptive quantization, adaptive equalization, digital filtering, and modulation and coding for voiceband data modems.

Dr. Gersho has 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, the Data Communications Technical Committee, and the Signal Processing and Communications Electronics Technical Committee of the IEEE Communications Society. He has served as Editor of IEEE COMMUNICATIONS MAGAZINE and as Associate Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS. He is also a member of the Task Force on Neural Networks for the IEEE Computer Society. 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 he received a NASA award for technical innovation.