Fixed-rate Entropy-coded Vector Quantization Using Linear (Zero-one) Programming

A. K. Khandani
Dept. of Elec. and Comp. Eng., University of Waterloo, Waterloo, Ont., N2L 3G1

Abstract[1]*:* Consider two sets of points X, A and their n-fold cartesian products $\{A\}^n$, $\{X\}^n$. A non-negative cost is associated with each element of A. A measure of distance is defined between an element of A and an element of X. It is assumed that the cost and also the distance in the n-fold space has an additive property. The shaped set is composed of a subset of elements of $\{A\}^n$ of the least cost. Decoding of an element $\mathbf{x} \in \{X\}^n$ is the process of finding the element of the shaped set which has the minimum distance to $\mathbf{x}$. Using the additivity property of cost and distance measures, the decoding problem is formulated as a linear program. Using the generalized upper bounding technique of linear programming in conjunction with some special features of the problem, we present methods to substantially reduce the complexity of the corresponding simplex search. The proposed method is used for the fixed-rate entropy-coded vector quantization of a Gaussian source. For $n = 128$ (space dimensionality) using 8 points per dimension and for a rate of 2.5 bits/dimension, we need about 52 additions, 87 comparisons, 0.2 divisions, and 0.4 multiplications per dimension to achieve SNR = 13.31 dB (the bound obtained from the Rate-distortion curve is 13.52 dB). This is substantially less complex than the traditional methods based on the dynamic programming.

1 Introduction

Consider a discrete set of points A composed of $K = |A|$ elements. A non-negative cost $c(a)$ is associated with each element $a \in A$. Consider the n-fold cartesian product of A dented as $\{A\}^n$. It is assumed that the cost of an n-fold element $\mathbf{a} = (a_0, \ldots, a_{n-1}) \in \{A\}^n$ is equal to: $c(\mathbf{a}) = \sum_i c(a_i)$. Shaping is achieved by selecting a subset of the n-fold elements, $S_n \in \{A\}^n$, with a cost less than or equal to a given value $c_{\max}$. We refer to A as the *constituent subset* and to S_n as the *shaped set.*

Two major applications of this problem are in: (i) quantizer shaping (fixed-rate entropy-coded vector quantization) of an independent source [1], [2] where A is the set of reconstruction vectors of an ℓ-D vector quantizer (for a given ℓ) and cost is equivalent to self-information, and, (ii) constellation shaping [3], [4] where A is the set of points of an ℓ-D subconstellation and cost is equivalent to energy.

Consider another set of n-tuples X_n denoted as the *input set.* A non-negative distance is defined between each $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in X_n$ and each $\mathbf{s} = (s_0, \ldots, s_{n-1}) \in S_n$. The distance between x_i and s_i is denoted as $d(x_i, s_i)$. It is assumed that the distance between $\mathbf{x}$ and $\mathbf{s}$ is equal to: $d(\mathbf{x}, \mathbf{s}) = \sum_i d(x_i, s_i)$. Decoding of an element $\mathbf{x} \in X_n$ is to find the element $\mathbf{s} \in S_n$ which has the minimum distance to $\mathbf{x}$. Selecting a point with a larger distance than the minimum possible value results in some degradation in performance.

The immediate approach to decoding is to perform an exhaustive search. This is achieved by computing the distance of $\mathbf{x}$ to all the elements of S_n and finding the smallest value. In most cases S_n has a huge cardinality and, consequently, the exhaustive search is impractical. In this case, one needs an algorithmic approach for decoding. The basic frameworks to develop such an algorithm are: (i) additivity property of cost, and, (ii) additivity property of distance.

A class of decoding schemes are based on replacing the shaping region by the Voronoi region of a lattice [5]. These schemes do not rely on the two additivity properties mentioned earlier. Another known approach for decoding is based on dynamic programming [6], [7], [8]. In this case, the states of the system correspond to the accumulative cost. One major problem associated with this approach is that the number of states can be a huge value. An effective rule for the quantization of the state space (merging of states) is presented in [8]. This results in a suboptimum shaping region with low decoding complexity and small performance degradation. Although for moderate values of n (say $n = 32$) this approach is quite effective, for larger values of n the decoding complexity to keep the performance degradation small still remains quite high.

In this paper, we introduce a method based on a linear programming approach which makes use of the additivity properties mentioned earlier in a more useful way. Decoding is achieved in a number of steps where each step finds a point of the shaped set with a smaller distance. This property enables us to provide a tradeoff between the search complexity and the performance.

This is the first time that the problem of decoding of a shaped set, and in specific the problem of fixed-rate

[1]This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC).

entropy-coded vector quantization, is formulated in terms of a linear program. This formulation enables us to apply the rich ideas developed in various contexts of linear programming to this new application.

In most parts of the paper, it is assumed that the reader is familiar with the general idea of linear programming and knows the details of the technique developed in [9].

2 Formulation of decoding as a linear program

A linear program is an optimization problem involving a linear objective function and also linear constraints. The basic theorem of linear programming says that in a problem composed of M equality constraints[2], the optimum answer is composed of M nonzero components. Any solution with M nonzero components, denoted as *basic variables* or *basis*, and satisfying the set of the constraints is called a *basic feasible* solution. This theorem also gives the necessary and sufficient conditions for a basic feasible solution to be optimum. Simplex method is a systematic search procedure which searches among the basic feasible solutions and finds the optimum answer in a finite number of steps. The search is composed of a set of pivoting operations where each such operation brings one of the nonbasic variables into the basis and replaces it with a basic variable. This is achieved such that the corresponding change in the objective function is maximized.

To formulate the decoding problem as a linear program, the elements of the ith constituent subset are identified by the use of a K-D binary vector $\{\delta_i(j), j=0,\ldots,K-1\}$ where $\delta_i(j)=0,1$ and $\sum_j \delta_i(j)=1,\ i=0,\ldots,n-1$. The vector corresponding to the jth element is composed of a single one in the jth position and zeros elsewhere.

The cost associated with the jth element of A is denoted as $c(j)$. For an n-tuple input $\mathbf{x}$, the distance of the ith component of $\mathbf{x}$ to the jth element of A is denoted as $d_i(j)$. The overall distance and cost are equal to: $\sum_i \sum_j \delta_i(j)d_i(j)$ and $\sum_i \sum_j \delta_i(j)c(j)$, respectively. The optimization problem is formulated as:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=0}^{n-1}\sum_{j=0}^{K-1} \delta_i(j)d_i(j) \\ \text{subject to:} \quad & \sum_{i=0}^{n-1}\sum_{j=0}^{K-1} \delta_i(j)c(j) + s_c = c_{\max}, \\ & \delta_i(j) = 0,1, \quad \forall i,j \quad \& \quad \sum_j \delta_i(j) = 1, \quad \forall i, \end{aligned} \tag{1}$$

where s_c is the slack variable of the cost constraint. Each of the equalities $\sum_j \delta_i(j)=1,\ i=0,\ldots,n-1$, is called an *indicator constraint.*

[2]In the case of an inequality constraint, by introducing an extra variable for each such constraint (denoted as the *slack variable*), it is transformed into the form of an equality.

The immediate problem in applying the simplex method to solve (1) is that the variables $\delta_i(j)$ are restricted to be integer numbers, or more specifically 0 and 1. In the context of linear programming, this is called a zero-one program. Fortunately, in the present case, one can avoid this complexity using the following simple theorem:

Theorem: There are at least one and at most two basic variables corresponding to each indicator constraint.

Proof: To satisfy the equality, there should be at least one basic variable corresponding to each indicator constraint. Considering that: (i) each indicator constraint is composed of a disjoint set of variables, and, (ii) the number of basic variables is equal to the number of indicator constraints plus one (because we have just one extra constraint namely the cost constraint), we conclude that there can not be more than two basic variables corresponding to each of them. An indicator constraint with two basic variables is called an *essential constraint.* The theorem says that the number of essential constraints is either zero or one.

The standard form for the resulting problem is given in (4). The constituent subsets are indexed by $0,\ldots,n-1$ (*set-index*). The set-indices are shown in the upper row of (4). The subset indexed by n will be defined later. The rows in (4), which are indexed by $0,\ldots,n+1$ (*row-index*), are generally denoted as the *equality constraints.* The row-indices are shown in the first column of (4). Each of the basic variables (in a given iteration of the simplex search) corresponds to one of the equality constraints. The indicator constraints are a subset of the equality constraints indexed by $2,\ldots,n+1$.

To solve the problem in (4), we just relax the zero-one constraint and then apply the simplex search. Using the previous theorem, we conclude that in the final solution at most two basic variables are different from unity. One of the following two cases may happen in the final answer:

Case I: If there is only one non-unity basic variable, it corresponds to the slack variable of the cost constraint and indicates that the cost constraint is satisfied with inequality. In this case, the vector obtained by concatenating the nearest points of different constituent subsets satisfy the cost constraint and is the optimum answer. By performing a simple test before starting the search procedure, one can avoid the computation associated with such cases.

Case II: If there are two non-unity basic variables, it means that the cost constraint is satisfied with equality. In this case, we set one of the non-unity basic variables to zero and the other one to unity. The selection is achieved such that the cost constraint is not violated. Obviously, such a selection is always possible and it can be shown that the resulting solution in indeed the optimum answer subject to the zero-one constraint [11].

Using the revised simplex method to solve the linear program given in (4) results in a basis of size $(n+2)\times(n+2)$. The problem has a special structure which can be used to substantially reduce the effective size of the basis. This is due to the fact that the set of the indicator constraints are non-overlapping. To use this structure, one basic variable in each indicator constraint is denoted as the *key variable.* If there are two basic variables (corresponding to an essential constraint), one of them is selected arbitrarily. The basic variable of an essential constraint which is not key is called the *non-key basic variable.* In the following, we see how one can perform the steps of the revised simplex search without considering the effect of the key variables in an explicit way.

Consider the system obtained by subtracting the column corresponding to each key variable from every other column in their respective set. This operation is equivalent to a linear change of variable and results in the so-called *derived system.* It can be easily shown that in this new system, the values of the key variables corresponding to any feasible solution is equal to one. These variables are treated as the variables at upper-bound in an upper bounded variable algorithm for the revised simplex method. In this way, one can assume that the key variables are absent from the derived system. This results in a *reduced system.* It can be shown that the basis for the reduced system, denoted as $\mathbf{B}$, is composed of the basic variables which are not key. The main idea is that the steps of the revised simplex search can be carry out using just the inverse $\mathbf{B}^{-1}$ of $\mathbf{B}$ and the corresponding basic solution of the reduced system. This property is due to the fact that one can easily reach from the basis of $\mathbf{B}$ to the corresponding basis in the unreduced system (denoted as the *unreduced basis* of $\mathbf{B}$). We just explain the operation of switching the basic variables.

First of all, it is easy to show that the variables to be switched can not belong to two different non-essential indicator constraints. If the variables belong to the same non-essential indicator constraint, we just change the corresponding key variable and do some necessary updating (no pivoting is required). A more complicated case occurs when the switching involves an essential constraint. In this case, if the switching is to be done with the corresponding non-key basic variable, we just perform the ordinary pivoting operation. But, if the switching is to be done with the key variable, we first change the key variable with the non-key basic variable, do some necessary updating, and then perform the pivoting operation.

In the following, we have some definitions which facilitate formulation and also implementation of the algorithm.

- Define $\mathcal{K}(i) \in [0, K-1]$ to contain the index of the key variable corresponding to the ith indicator constraint, $i = 0, \ldots, n-1$.
- Define the vector of variables $\mathbf{v}$ as:

  $v(iK+j) = \delta_i(j)$ for $i = 0, \ldots, n-1$, $j = 0, \ldots, K-1$, and $v(nK) = s_c$.
- Define $\mathcal{B}(i)$ to contain the index (in $\mathbf{v}$) of the basic variable corresponding to the ith equality constraint, $i = 1, \ldots, n+1$,[3].

Our problem has two special features with respect to the general case discussed in [9] which are used to reduce the computational complexity. These are explained in the following:

- The costs of points for all the constituent subsets are the same.
- There is only one constraint involving all the variables (cost constraint). As already mentioned, the immediate consequence of this feature is that one can relax the zero-one constraint. In addition, this results in a matrix $\mathbf{B}$ of size 2×2 which is upper triangular (with a unity element at the upper left corner).

The corresponding algorithm is explained in the next section. The (i, j)th element of $\mathbf{B}^{-1}$ is denoted as $B^{-1}(i, j)$.

3 Algorithm

1. Start from the basic feasible solution where the selected component for all the constituent subsets is the point of the least cost.
2. Compute $\phi(j) = B^{-1}(0,1)c(j)$, $\varphi(j) = B^{-1}(1,1)c(j)$ for $j = 0, \ldots, K-1$.
3. Compute the 2-D vector $\bar{\mathbf{b}}$ where $\bar{b}(0) = B^{-1}(0,1)c_{\max}$, $\bar{b}(1) = B^{-1}(1,1)c_{\max}$.
4. Compute $\mu_i(j) = d_i(j) + \phi(j)$ for $i = 0, \ldots, n-1$, $j = 0, \ldots, K-1$. This is the inner product of the first row of $\mathbf{B}^{-1}$ with the vector composed of the first two elements in each column of the system given in (4).
5. For each value of the set-index $i = 0, \ldots, n-1$, find the index $\omega_i \in [0, K-1]$ such that $\mu_i(\omega_i) = \min_j \mu_i(j)$.
6. Compute $\Delta_i = \mu_i(\omega_i) - \mu_i[\mathcal{K}(i)]$ for $i = 0, \ldots, n-1$.
7. Compute the 2-D vector $\bar{\mathbf{d}}$ with the components $\bar{d}(0) = \bar{b}(0) - \sum_{i=0}^{n-1} \mu_i[\mathcal{K}(i)]$, $\bar{d}(1) = \bar{b}(1) - \sum_{i=0}^{n-1} \varphi[\mathcal{K}(i)]$.
8. Find the set-index σ such that $\Delta_\sigma = \min_i \Delta_i$, $i = 0, \ldots, n$, and assume that the variable resulting in $\min_j \mu_\sigma(j)$ in step 5 is indexed (in $\mathbf{v}$) by s.
9. If $\Delta_\sigma \geq 0$, the optimum solution is found. Otherwise, bring the variable indexed by s into the basis. This is achieved in the following:
10. Compute $\Gamma_\sigma = \varphi(\omega_\sigma) - \varphi[\mathcal{K}(\sigma)]$.
11. Set $\overline{D}(0) = \Delta_\sigma$, $\overline{D}(1) = \Gamma_\sigma$, and $\Delta_\sigma = 0$.

[3]The basic variable corresponding to the equality constraint indexed by $i = 0$ is always z_0 which is not considered here.

12. Expand $\overline{\mathbf{D}} = [\overline{D}(0), \overline{D}(1)]$ on the original basis. The result is denoted as $\mathbf{A}_s$.

13. Expand $\overline{\mathrm{d}}$ on the original basis. The result is denoted as b.

14. Compute the index r of the variable to leave the basis using,

$$r = \mathcal{B}(\psi) \quad \text{where} \quad \frac{b(\psi)}{A_s(\psi)} = \min_{A_s(i)>0} \frac{b(i)}{A_s(i)}. \tag{2}$$

Assume that this variable belongs to the constituent subset indexed by ρ.

15. If $\rho = \sigma$ and the corresponding constraint is not essential, change the key variable of the constraint and perform the necessary updating.

16. If the variable to leave the basis is the key variable of an essential constraint, make the corresponding non-key basic variable to be the key and perform the necessary updating.

17. Perform the pivoting operation by replacing $\mathbf{B}^{-1}$ by $\mathbf{PB}^{-1}$ where,

$$\mathbf{P} = \begin{bmatrix} 1 & -\frac{\overline{D}(0)}{\overline{D}(1)} \\ 0 & \frac{1}{\overline{D}(1)} \end{bmatrix}, \tag{3}$$

and perform the necessary updating.

The algorithm can be easily modified to provide a trade-off between the search complexity and performance. One way to do this is to impose a constraint on the total execution time.

A more detailed explanation of this algorithm is available in [11].

Note: In some applications of shaping, the constituent subsets are not the same. An example of this case is in the vector quantization (in the transform domain) of a correlated source [10]. The algorithm given here can be easily modified to apply to that more general problem.

4 Numerical Results

The algorithm given in section 3 is applied to the quantization of a Gaussian source. The results are verified against the results obtained using a general purpose linear programming package. The values of n up to 512 are tested. For $n = 512$, the measured SNR is only 0.11 db below the bound determined by the Rate-distortion curve of a Gaussian source. In general, the number of iterations to reach the nearest point or very close to it is quite small (in the order of a few tens). The majority of the iterations do not need pivoting. The overall complexity is substantially lower than the complexity of the methods based on dynamic programming as discussed in [6], [7], [8]. At the same time, the performance is better because: (i) no quantization of the state space is involved, and, (ii) one can use much larger values of n.

As a specific example, for $n = 128$ using 8 points per dimension and for a rate of 2.5 bits/dimension, we need about 52 additions, 87 comparisons, 0.2 divisions, and 0.4 multiplications per dimension to achieve SNR = 13.31 dB (the bound obtained from the Rate-distortion curve is 13.52 dB). As a matter of comparison, in the dynamic programming approach, by quantizing the self-information of the points along each dimension to 256 (or 128) different values, we obtain SNR = 13.25 (or 13.21) dB and the complexity per dimension of the decoder is equivalent to the Viterbi decoding of a trellis with about 3650 (or 900) states. This requires about 14600 (or 3600) addition, 10950 (or 2700) comparison per dimension. In addition to this large computational complexity, the method based on dynamic programming also needs a large amount of RAM memory to keep track of the surviving paths through the trellis. In this specific example, this add up to about 467200 (or 115200) words of RAM memory. Note that in the proposed method the RAM memory is mainly used to store the distances and is equal to $nK/2$ words of memory. For the example discussed here ($n = 128, K = 8$), this is only 256 words of RAM memory.

As another example, for $n = 512$, using 8 points per dimension and for a rate of 2.5 bits/dimension, we need about 166 additions, 342 comparisons 0.2 divisions, and 0.4 multiplications per dimension to achieve SNR = 13.41 dB (the bound obtained from the Rate-distortion curve is 13.52 dB).

Extensive numerical results concerning the performance and the complexity of the proposed quantization scheme are available in [11]. Some examples of such numerical results are given in Fig. (1) which show the performance and the complexity of the proposed coding scheme as a function of dimension.

References

[1] D. J. Sakrison, "A geometrical treatment of the source encoding of a Gaussian random variable," *IEEE Trans. Inform. Theory,* vol. IT-14, pp. 481–486, May 1968.

[2] T. R. Fischer, "Geometric source coding and vector quantization," *IEEE Trans. Inform. Theory,* vol. IT-35, pp. 137–145, Jan. 1989.

[3] G. D. Forney, Jr. and L. F. Wei, "Multidimensional constellations–Part I: Introduction, figures of merit, and generalized cross constellations," *IEEE J. Select. Areas Commun.,* vol. SAC-7, pp. 877–892, Aug. 1989.

[4] A. R. Calderbank and L. H. Ozarow, "Nonequiprobable signaling on the gaussian channel," *IEEE Trans. Inform. Theory,* vol. IT-36, pp. 726–740, July 1990.

[5] M. V. Eyuboglu and G. D. Forney, "Lattice and trellis quantization with lattice- and trellis-bounded codebooks—high-rate theory for memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-39, pp. 46–59, Jan. 1993.

[6] R. Laroia and N. Farvardin, "A structured fixed-rate vector quantizer derived from variable-length scalar quantizer—Part I: Memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-39, pp. 851–867, May 1993.

[7] A. S. Balamesh and D. L. Neuhoff, "Block-constrained methods of fixed-rate, entropy coded, scalar quantization," submitted to *IEEE Trans. Inform. Theory,* Sept. 1992.

[8] A. K. Khandani, "A dynamic programming approach to fixed-rate entropy-coded vector quantization," submitted to *IEEE Trans. Inform. Theory.*

[9] G. B. Dantzig, and R. M. Van Slyke, "Generalized upper bounding techniques," *Journal of Computer and System Sciences,* pp. 213–226, 1967.

[10] D. G. Jeong and J. D. Gibson, "Uniform and piecewise uniform lattice vector quantization for memoryless gaussian and laplacian sources," submitted to *IEEE Trans. Inform. Theory,* Jan. 1991, revised March 1992.

[11] A. K. Khandani, "Using simplex search to decode a cartesian product set with a constraint on an additive cost — Fixed-rate entropy-coded vector quantization," submitted to *IEEE Trans. Inform. Theory.*

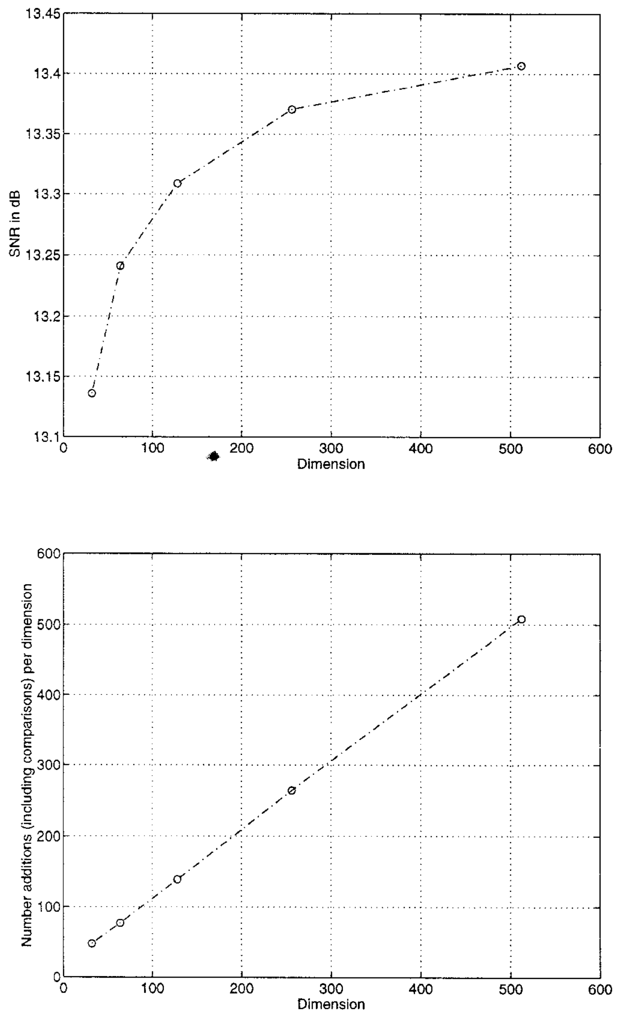

Figure 1: SNR (in dB) and complexity (in number of addition, comparisons per dimension) of the proposed quantization scheme in conjunction with a Gaussian source. Number of multiplication (including divisions) per dimension is about 0.6 for $n = 32, \dots, 512$. Number of points per dimension is 8 and rate is 2.5 bits/dimensions. The bound on SNR obtained from the Rate-distortion curve is 13.52 dB.

Maximize $z_0 = -\sum_i \sum_j \delta_i(j) d_i(j)$, subject to:

(4)

$$\begin{array}{r|lllllll}
 & 0 & & \cdots\cdots & n-1 & & n & \\ \hline
0 & \delta_0(0)d_0(0) & + \dots \delta_0(K-1)d_0(K-1) + & \cdots\cdots \delta_{n-1}(0)d_{n-1}(0) & + \dots \delta_{n-1}(K-1)d_{n-1}(K-1) & + z_0 & & = 0 \\
1 & \delta_0(0)c(0) & + \dots \delta_0(K-1)c(K-1) + & \cdots\cdots \delta_{n-1}(0)c(0) & + \dots \delta_{n-1}(K-1)c(K-1) & & + s_c & = c_{\max} \\
2 & \delta_0(0) & + \dots \delta_0(K-1) & & & & & = 1 \\
 & & & & & & & \\
\cdot & & & \cdots\cdots & & & & \\
 & & & & & & & \\
n+1 & & & \delta_{n-1}(0) & + \dots \delta_{n-1}(K-1) & & & = 1
\end{array}$$