A Low Complexity Method for Fixed-rate Entropy-constrained Vector Quantization

S. Nikneshan and A. K. Khandani

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

October 28, 2005

Abstract: This paper describes a new approach to Fixed-rate Entropy-constrained Vector Quantization (FEVQ) for stationary memoryless sources where the structure of code-words are derived from a variable-length scalar quantizer. We formulate the quantization search operation as a zero-one integer optimization problem [1], and show that the resulting integer program can be closely approximated by solving a simple linear program. The result is a Lagrange formulation which adjoins the constraint on the entropy (codeword length) to the distortion. Unlike the previously known methods with a fixed Lagrange multiplier, we use an iterative algorithm to optimize the underlying objective function while updating the Lagrange multiplier until the constraint on the overall rate is satisfied. The key feature of the new method is the substantial reduction in the number of iterations in comparison with previous related methods [2]. In order to achieve some packing gain, we combine the process of Trellis Coded Quantization (TCQ) with that of FEVQ. This results in an iterative application of the Viterbi algorithm on the underlying trellis for selecting the Lagrange multiplier. Numerical results are presented which demonstrate substantial improvement in comparison with the alternative methods reported in the literature.

1 Introduction

Optimum fixed-rate scalar quantizers, introduced by Max [3] and Lloyd [4] (Lloyd-Max Quantizer or LMQ), minimize the average distortion for a given number of threshold points. In spite of the gain of LMQ in comparison with uniform quantization, there is a big gap between the LMQ performance and the rate distortion bound due to the lack of entropy coding.

An extension of LMQ to vector sources introduced by Linde, Buzo, and Gray in [5], performs arbitrarily close to the rate-distortion bound as the space dimension becomes larger. However, the complexity of this method is exponential in NR where N is the dimension and R is the per sample rate, making it impractical even for modest values of rate and dimension.

To improve the performance of scalar quantizers, one could use variable-length encoding (entropy coding) of the quantizer output. The general design methodology for such an entropy-coded quantizer is based on the minimization of the functional $J = D + \lambda H$ where D is the quantization distortion, λ is the Lagrange multiplier, and H is the entropy of the output. This type of optimization problem in Information Theory was first presented by Blahut [6] to compute the rate-distortion function. A well known method for entropy-coded quantization, introduced in [7], is based on using a distance measure which combines the effects of entropy and distortion by considering a linear combination of the two using a fixed Lagrange multiplier. The quantizer design in [7] is based on a variation of the Lloyd-Max algorithm. The current work takes advantage of a similar Lagrangian formulation composed of a linear combination of the code-word length and the quantization distortion. We use an iterative algorithm to change the Lagrange multiplier until a certain constraint on the overall rate is satisfied. The algorithm updates the Lagrange multiplier such that the changes in the objective function value at each step is maximized. This results in a chain of solutions which provides the maximum possible improvement at each step. Numerical comparisons are presented versus other methods suggested for the iterative selection of the Lagrange multiplier [2] which demonstrate a significant reduction in the complexity (measured in terms of the required number of iterations).

To take advantage of entropy coding, while avoiding the disadvantages associated with conventional methods based on using variable rate codes including error propagation and buffering problems, one can use Fixed-rate Entropy-constrained Vector Quantization (FEVQ). This is based on selecting a subset of points of high probability in the Cartesian product of a set of scalar quantizers and representing their elements with binary code-words of the same length. In this case, as some of elements are discarded from the Cartesian product space, the operations of "nearest neighbor search" and "labeling of quantizer points (addressing)" can no longer be achieved independently along the space dimensions. The main challenge here would be to find methods to exploit the structure of the selected subset of points to reduce the complexity of these operations. One can further improve the performance of FEVQ by exploiting a high

dimensional lattice which offers some extra quantization (packing) gain by using quantization regions which are closer to spherical as compared to cubes.

The pyramid vector quantizer, introduced by Fischer (for Laplacian sources) [8], is a fixed-rate VQ in which the code-vectors are located on the intersection of a cubic lattice and a pyramid in N-dimensional space.

One class of FEVQ schemes is based on using a subset of points from a lattice bounded within the Voronoi region around the origin of another lattice (shaping lattice) [9]. This approach has been extended in [9] to the case of using the trellis diagram of a convolutional code to construct the shaping lattice. In both cases, the selected subset forms a group under the vector addition modulo the shaping lattice. This group property is used to facilitate the underlying operations ("nearest neighbor search" and "labeling of quantizer points").

Another class of schemes is based on selecting the N-dimensional quantizer points with the lowest additive self-information (typical set). In this case, the selected subset has a high degree of structure which can be used to reduce the complexity. A method for exploiting this structure, based on using dynamic programming is proposed by Laroia and Farvardin in [10]. The core idea in the schemes of [10] is to use a state diagram with the transitions corresponding to the one-dimensional symbols. This results in a trellis composed of N stages where N is the space dimensionality. The states s and $s+c$ in two successive stages are connected by a link corresponding to the one-dimensional symbol(s) of cost c. Consequently, the states in the nth stage, $n=0,\ldots,N-1$, represent the accumulative cost over the set of the first n dimensions. The links connecting two successive stages are assigned a metric corresponding to the one-dimensional symbol distortions. Then, the Viterbi algorithm is used to find the path of the minimum overall additive distortion through the trellis. In [10], cost is defined as a scaled version of self information where entropy coding is achieved by discarding the states with a cost higher than a given threshold. This Scalar-Vector Quantizer (SVQ) [10] is a vector quantizer derived from a variable-rate scalar quantizer which can achieve a large portion of the boundary gain by selecting the code-vectors inside the typical set. However, no packing gain is realized by SVQ as the underlying grid is rectangular.

Reference [11] uses a different approach to dynamic programming showing improvement with respect to the schemes of [10]. The key point in [11] is to decompose the underlying operations into the lower dimensional subspaces. This decomposition avoids the exponential

growth of the complexity. The core of the scheme, as in any problem of dynamic programming, is a recursive relationship which is formed in a hierarchy of levels where each level involves the Cartesian product of two lower dimensional subspaces.

Reference [2] presents three methods to reduce the search complexity in Fixed-Rate Entropy-constrained Scalar Quantization: "greedy search", "differential search", and "Lagrange-multiplier-based search". These are briefly explained below and later, in the numerical result section, we present a comparison with the "Lagrange-multiplier-based search" as it is claimed in [2] to provide the lowest complexity and the least degradation in performance in comparison with the other methods.

In "greedy search" [2], the authors start from the unconstrained quantizer output and replace some of its components with a quantization level with a smaller length. This is achieved such that the reduction in the code-word length per unit distortion increase is maximized. Independent of [2], a method based on a similar principle is presented in our earlier work [12][1] where the focus in [12] has been on deriving sufficient conditions for the optimality of such a greedy search. The algorithm suggested in [2] does not allow each component to be perturbed more than once and continues until the cost constraint is satisfied. The greedy algorithm in [2] may fail to produce a valid output in which case another simpler algorithm is used. This method reduces the complexity, but it requires much higher block lengths (larger delay) to produce a performance close to the results reported in the current article.

In "differential search" [2], the authors limit the number of states at each stage of the dynamic programming search. The key idea is that the states in the optimal path through the trellis are close to the states of the unconstrained quantizer output when the dynamic programming approach is used. The differential search reduces the complexity at the cost of some degradation in performance.

In "Lagrange-multiplier-based search" [2], motivated by the work of Chou *et al.* [7], the authors minimize the linear combination of distance and total length ($D + \lambda L$). The algorithm increases the Lagrange multiplier gradually from zero until the cost constraint is satisfied. This method is based on a similar principle as the method proposed here, but unlike the current article, the authors in [2] do not present any technique to reduce the number of iterations by

[1] We became aware of the similarity between [12] and [2] after [12] was published.

optimizing the variation of the Lagrange multiplier. The authors in [2] mention the possibility of using a bisection method on λ for this purpose, but do not present any numerical results. We have implemented such a bisection method on λ for the sake of comparison as will be reported later in the numerical result section.

The Trellis Coded Quantizer (TCQ) introduced by Marcellin and Fischer in [13] is also derived from a scalar quantizer and uses the Ungerboeck idea of coding by set partitioning to realize a significant quantization gain. It is based on applying the Ungerboeck notion of set partitioning to the partitions of a scalar quantizer where a trellis structure is used to prune the expanded number of quantization levels down to the desired encoding rate. Reference [13] studies the quantization gain of several Ungerboeck one-dimensional trellis codes based on partitioning the integer lattice Z into the four cosets of $4Z$. Following [13], several other variations of TCQ have been studied in the literature. Trellis Coded Vector Quantizer, studied in [14, 15], is based on applying the coding by set partitioning idea on a vector quantizer codebook.

The Entropy-coded TCQ of [16, 17] uses variable rate coding of the output. This is a generalization of the method of [7] (using fixed Lagrange multiplier[2]) to include TCQ. The Entropy-coded TCQ of Fischer and Wang [16] performs within 0.5 dB of the rate distortion bound. In spite of their excellent performance, the problems associated with variable-length coding limits the usefulness of these approaches in many practical situations.

In order to achieve some boundary gain, Laroia and Farvardin combine a scalar-vector quantizer [10] with trellis coded quantization [13] and propose a new fixed-rate scheme, known as Trellis-Based Scalar-Vector Quantizer (TB-SVQ) [18]. The resulting quantizer shows an excellent performance assuming error free transmission. However, as shown in [19], TB-SVQ is a catastrophic code, and a single bit error within a block can propagate indefinitely into other blocks. In [19], Yang and Fischer propose a solution to this problem in which the channel error can propagate two blocks at most.

In this paper, we introduce a low complexity method for FEVQ for stationary memoryless sources. This algorithm uses a zero-one integer optimization formulation for the nearest neighbor search which is subsequently approximated by a simpler linear program [1]. We propose a

[2]This is based on using a distance measure which is a linear combination (with fixed coefficients) of the codeword length and quantization distortion.

new solution for this formulation which is less complex when compared to the solution proposed in [1], and also, unlike the method of [1], it can be easily combined with a quantization lattice (in specific a TCQ) to achieve some packing gain. By applying the Dantzig-Wolfe decomposition, we break the linear program into smaller sub-problems that can be easily solved by iterating between the sub-problems. We show that the optimum solution of the linear program results in a Lagrangian formulation adjoining the distortion and the length of the codewords. We show that finding the optimum solution of the underlying linear program is equivalent to finding the optimum value for the corresponding Lagrange multiplier. Starting from two initial points, we derive a method to find a chain of solutions which provides the maximum possible improvement at each step. The new method reduces the number of iterations to reach to the optimum point in comparison with that of reference [2]. We also show that the new method converges more quickly than a numerical bisection. That makes the proposed method more attractive for real time applications where the complexity and delay are important issues. As already mentioned, the approach can be easily mixed with trellis/lattice quantization by an iterative application of the proposed algorithm to the trellis diagram of the underlying lattice (using Viterbi algorithm).

The rest of paper is organized as follows: Section 2 contains a brief description of the linear programming formulation and the approach to solve this problem. We also discuss a method to combine the proposed approach to FEVQ with trellis quantization (TCQ). Finally, in Section 3, we conclude the paper by presenting the numerical results and a comparison between the proposed methods with some other quantization schemes.

2 Formulation of Nearest Neighbor Search as a Linear Program

Consider an N dimensional vector quantizer derived from N variable length scalar quantizers. Each scalar quantizer consists of M partitions with reconstruction levels $q_1, q_2, ..., q_M$ where $q_1 < q_2 < ... < q_M$. There is a variable-length, binary, prefix code associated with each quantizer partition.

There is a set of positive cost values $c(1), c(2), ..., c(M)$ associated with the quantizer par-

titions where the constraint on the entropy is formulated as the sum of the cost values along different dimensions being less than a given threshold, say $C_{\max}$. A natural choice is to use the self information associated with the quantizer partitions as their respective cost value. In this case, the constraint on the overall cost results in selecting a subset of points of the highest probability in the Cartesian product space. In [1, 10, 11, 18], the cost is obtained by scaling the self information. In [10, 18], it is assumed that the cost values $c(1), c(2), ..., c(M)$ and $C_{\max}$ are integers[3]. Using integer cost values is the basis behind constructing the trellis diagram in [10, 18] which is used for "nearest neighbor search" and "labeling" of the quantizer partitions. The method proposed here can be applied to arbitrary cost values. However, to facilitate the addressing, the cost values are simply defined as the length of the binary codewords associated with the partitions. This corresponds to a coarse approximation of the self-information values and results in a loss in the achievable performance. We show that the corresponding degradation in the performance can be reduced by merging several dimensions together and applying the binary labels to the points in the resulting subspace. One could also define the cost values similar to [10, 18] and use the algorithm proposed here for the "nearest neighbor search" and the enumeration algorithm of [10] for the labeling.

To formulate the nearest neighbor search problem, the jth partition of the scalar quantizer along the ith dimension is identified by a binary variable $\delta_i(j)$, $i = 1, ..., N$, $j = 1, ..., M$ where $\delta_i(j) \in \{0, 1\}$ and $\sum_j \delta_i(j) = 1$, $i = 1, ..., N$. To select the element indexed by j_0 along the ith dimension, we set $\delta_i(j_0) = 1$ and $\delta_i(j) = 0$, $j \neq j_0$. Given an input vector $\mathbf{r} = (r_1, \ldots, r_N)$, the quantization problem (search for the nearest neighbor) is formulated as,

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{N} \sum_{j=1}^{M} \delta_i(j) d_i(j) \\ \text{Subject to} \quad & \sum_{i=1}^{N} \sum_{j=1}^{M} \delta_i(j) c(j) \leq C_{\max}, \\ & \textstyle\sum_{j=1}^{M} \delta_i(j) = 1, \quad \forall i, \qquad \delta_i(j) = 0, 1, \quad \forall i, j, \end{aligned} \tag{1}$$

where $d_i(j) = (r_i - q_j)^2$. Each of the equalities $\sum_j \delta_i(j) = 1$, $i = 1, ..., N$, is called an *indicator constraint*.

Note that one can further reduce the complexity of the search algorithm by focusing on a

[3]Note that this assumption does not result in any loss of generality because non-integer values can be approximated (as closely as desired) by integers through applying a proper scaling and rounding of values.

subset of quantizer points along each dimension which can potentially become a component in the final solution. We refer to these subsets as the *candidate sets* corresponding to each dimension. To explain this idea, let us assign a second set of indices to the quantizer partitions along each dimension to index the elements within each candidate set. We assume that the elements of each candidate set are ordered according to their distance from the corresponding input component with the nearest point indexed by zero. Using these notations, the candidate set for the ith dimension is defined as the collection of quantizer partitions satisfying,

$$\tilde{d}_i(0) < \tilde{d}_i(1) \ldots < \tilde{d}_i(m_i)$$
$$\tilde{c}_i(0) > \tilde{c}_i(1) \ldots > \tilde{c}_i(m_i)$$

where $\tilde{d}_i(j)$ and $\tilde{c}_i(j)$ are the distance and the cost for the points of the i candidate set, respectively, and m_i is the cardinality of the ith candidate set ($m_i \leq M$, M is the number of threshold points along each dimension). This definition is justified noting that if two points α and β satisfy, $d_i(\alpha) \leq d_i(\beta)$ and $\tilde{c}_i(\alpha) \leq \tilde{c}_i(\beta)$, then α is always a better choice as compared to β, and consequently, β is not in the candidate set.

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. A general integer programming problem as well as a general zero-one programming problem is known to be NP-hard [20]. The available techniques to solve a general integer programming problem can be categorized into the following two major groups [20]: (i) cutting plane techniques, and (ii) enumerative techniques. There is also a fair amount of research on the specific topic of zero-one programming. Fortunately, in the present case, one can substantially reduce the search complexity using the following simple theorem which applies to a relaxed problem (removing the zero-one constraint) [1]:

Theorem 1: There are at least one and at most two non-zero variables corresponding to each indicator constraint[4].

To solve the problem in (1), we just relax the zero-one constraint and then find the optimum solution to the underlying linear program. Using the previous theorem, we conclude that in the relaxed solution of (1), at most two non-zero variables are different from unity. In the case where all the non-zero variables are equal to unity, it means that the solution of the

[4] A more general form of this theorem can be found in [21].

linear program coincides with the solution of the integer program and no rounding is needed. Otherwise, namely in the cases where there are two non-unity variables, we set one of the non-unity variables to zero and the other one to unity. The selection is achieved such that the cost constraint is not violated. It can be shown that such a selection is always possible [1]. This results in a valid, probably slightly sub-optimum solution. The resulting point is expected to be close to optimum, especially for large values of N, as the rounding is performed over at most one coordinate while the corresponding increase in the distortion is normalized by N. This results in a good overall performance for the proposed scheme as supported by numerical simulations.

In [1], a solution based on a generalized upper bounding technique is suggested to solve the underlying linear program. In the current article, we use an improved solution method with a lower complexity. In addition, the proposed solution has an interpretation in terms of the conventional Lagrangian method (where the corresponding Lagrange multiplier is iteratively optimized). This provides a natural framework to combine the method with a trellis structure to achieve some packing gain.

To find the optimum solution for the underlying linear program, we find a chain of solutions each expressed in terms of a linear interpolation between two intermediate points. The interpolation coefficients are computed such that the cost constraint is satisfied with equality. At each iteration, one of the two points involved in the interpolation is updated in a way that the resulting improvement in the objective function value is maximized. The updating is achieved by solving an LP problem which is solely subject to the indicator constraints and has a trivial complexity.

The proposed solution methodology is inspired by the Dantzig-Wolfe decomposition technique [22] for solving linear programs tailored to the structure given in (1). The Dantzig-Wolfe decomposition is a way of breaking a large linear programming problem into smaller subproblems that can be easily solved. A problem that is a good candidate for the Dantzig-Wolfe decomposition has the following three properties: there are several logical groups of variables; most constraints only affect one group of variables; and a small number of constraints link the groups of variables together. In equation (1), there are N indicator constraints each affecting one group of variables, and the cost constraint affecting all the variables. The key idea in the Dantzig-Wolfe decomposition is to reduce the complexity by iterating between two linear

programming sub-problems: one which is subject to the first set of constraints (the indicator constraint in our case), and the second one which is subject to the second set of constraints (cost constraint in our case). Search methods based on the Dantzig-Wolfe decomposition are known to be very effective, as by iterating between two simple sub-problems, the solution moves very quickly towards the global optimum point.

To proceed with the formulation, consider the solution $\mathbf{x} = \{x_i(j), i = 1, \cdots, N, j = 1, \cdots, M\}$ and assume that the objective function value and the cost associated with $\mathbf{x}$ are equal to,

$$\begin{aligned} D &= \sum_{i=1}^{N} \sum_{j=1}^{M} x_i(j) d_i(j) \\ C &= \sum_{i=1}^{N} \sum_{j=1}^{M} x_i(j) c(j), \end{aligned} \tag{2}$$

respectively. We look at any such solution $\mathbf{x}$ as providing a tradeoff between the cost value C and the objective function value D. The main idea is to find a linear interpolation between an appropriate set of such $\mathbf{x}$'s that: (i) the cost constraint is satisfied with equality, and (ii) the overall tradeoff between the cost value and the objective function value is optimized.

Consider a solution obtained by interpolating between two such points, say $\mathbf{x}^1$, $\mathbf{x}^2$, and assume that we try to improve the solution by bringing a third point, $\mathbf{x}^3$, into the interpolation procedure. The following LP problem is used to compute the interpolation coefficients for the selection of optimum $\mathbf{x}^3$,

$$\begin{aligned} \text{Minimize} \quad & \lambda_1 D_1 + \lambda_2 D_2 + \lambda_3 D_3 \\ \text{Subject to} \quad & \lambda_1 C_1 + \lambda_2 C_2 + \lambda_3 C_3 = C_{\max} \\ & \lambda_1 + \lambda_2 + \lambda_3 = 1 \end{aligned} \tag{3}$$

where D_1, D_2, D_3 and C_1, C_2, C_3 are the objective function and cost associated with points $\mathbf{x}^1$, $\mathbf{x}^2$, $\mathbf{x}^3$, respectively (note that $\lambda_1, \lambda_2, \lambda_3$ are the unknown variables). Without loss of generality, we assume that $C_1 < C_{\max} < C_2$ and $D_1 > D_2$. Using basic principles of linear programming, we note that as the LP problem in (3) has two constraints, only two of the corresponding λs will be non-zero. This simply means that it suffices to use only two points for the interpolation. This will be the case even if we try to provide an interpolation among a larger number of such points.

For a given value of λ_3, the LP problem in (3) is equivalent to:

$$\begin{array}{ll} \text{Minimize} & \lambda_1 D_1 + \lambda_2 D_2 + \lambda_3 D_3 \\ \text{Subject to} & \lambda_1 C_1 + \lambda_2 C_2 = C_{\max} - \lambda_3 C_3 \\ & \lambda_1 + \lambda_2 = 1 - \lambda_3 \end{array} \tag{4}$$

Solving for λ_1, λ_2, we obtain,

$$\lambda_1 = \frac{-C_{\max} + \lambda_3 C_3 + (1-\lambda_3) C_2}{C_2 - C_1} \quad \text{and} \tag{5}$$

$$\lambda_2 = \frac{C_{\max} - \lambda_3 C_3 - (1-\lambda_3) C_1}{C_2 - C_1}. \tag{6}$$

Substituting in (3) results in,

$$\begin{aligned} \lambda_1 D_1 + \lambda_2 D_2 &+ \lambda_3 D_3 = \lambda_3 (D_3 + \pi_1 C_3 + \pi_2) \\ &+ \frac{(C_2 - C_{\max}) D_1 + (C_{\max} - C_1) D_2}{C_2 - C_1} \end{aligned}$$

where,

$$\pi_1 = \frac{D_1 - D_2}{C_2 - C_1} \quad \text{and} \quad \pi_2 = \frac{C_1 D_2 - C_2 D_1}{C_2 - C_1} \tag{7}$$

The term $[(C_2 - C_{\max}) D_1 + (C_{\max} - C_1) D_2]/(C_2 - C_1)$ in (7) is the best value of the objective function obtained by interpolating between only $\mathbf{x}^1$ and $\mathbf{x}^2$. The optimum $\mathbf{x}^3$ is selected to minimize the effect of the related term $(D_3 + \pi_1 C_3)$ in (7). This results in the following LP problem for the selection of $\mathbf{x}^3$:

$$\begin{array}{ll} \text{Minimize} & D_3 + \pi_1 C_3 = \displaystyle\sum_{i=1}^{N} \sum_{j=1}^{M} \left[d_i(j) + \pi_1 c(j) \right] x_i^3(j) \\ \text{Subject to} & \text{Indicator constraints} \end{array} \tag{8}$$

where $x_i^3(j)$ are the components of $\mathbf{x}^3$. Note that the LP in (8) is decomposable, and consequently, has a trivial complexity. The corresponding solution is simply obtained by selecting the quantization partition along each dimension which minimizes the term $d_i(j) + \pi_1 c(j)$, namely the linear combination of the distance and cost obtained using the Lagrange multiplier π_1 (this is the quantization rule given in [7] for Lagrange multiplier π_1). If the minimum value of (8) satisfies $D_3 + \pi_1 C_3 + \pi_2 < 0$, it means that the inclusion of $\mathbf{x}^3$ results in a decrease in the objective function value in which case the iteration will continue. Then, one of the two points $\mathbf{x}^1$ or $\mathbf{x}^2$ is updated. In this update, $\mathbf{x}^3$ will replace $\mathbf{x}^1$, if $C_3 < C_{\max}$ and it will replace $\mathbf{x}^2$, if

$C_3 > C_{\max}$ (refer to Figure 1). In this manner, we always have $C_1 < C_{\max} < C_2$ and $D_1 > D_2$. After this, the whole procedure is repeated for the resulting two points until it merges to a stationary condition which, considering the linearity of the function, will be the global optimum solution of the linear program. The procedure is repeated (for I times) until no further change in the value of π_1 in two subsequent iterations is observed. Note that if $C_3 = C_{\max}$, the optimum solution is found and we stop. It should be also noted that if the unconstrained quantizer output satisfies the cost constraint, there is no need to run the algorithm and we have the optimum solution. If the total length of unconstrained output is less than $C_{\max}$, we use the padding technique to make it fixed-rate.

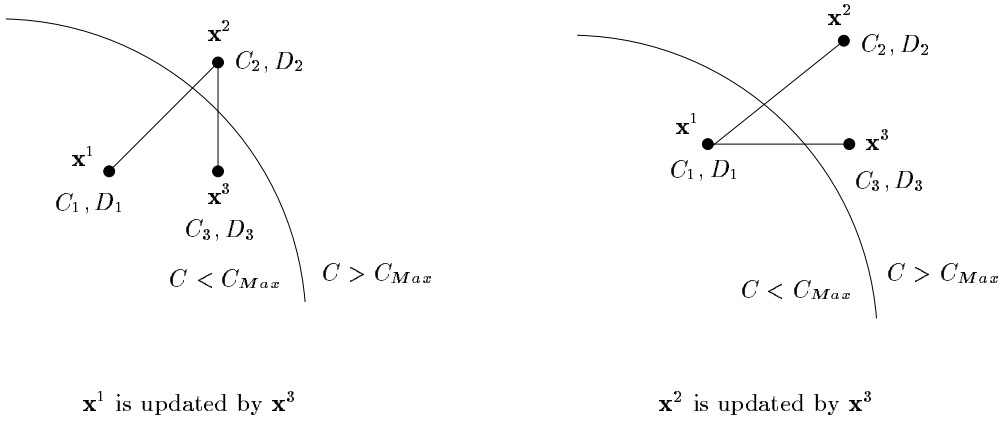

Figure 1: Updating of the points $\mathbf{x}^1$, $\mathbf{x}^1$ with a third point $\mathbf{x}^3$.

Relationship (8) has the following interesting interpretation. The point $\mathbf{x}^3$ is selected to minimize a linear combination of code-word length and distortion using a Lagrange multiplier π_1 which is the slope of the line between points (C_1, D_1) and (C_2, D_2) in the rate-distortion plane (corresponding to $\mathbf{x}^1$, $\mathbf{x}^2$, respectively). This feature enables us to explain the resulting iterative algorithm in terms of the following steps: (i) given $\mathbf{x}^1$, $\mathbf{x}^2$, compute π_1 as the slope of the line between (C_1, D_1), (C_2, D_2), (ii) find $\mathbf{x}^3$ using the quantization rule: Minimize $D_3 + \pi_1 C_3$ (this is the quantization rule given in [7] for Lagrange multiplier π_1), and (iii) replace $\mathbf{x}^1$ or $\mathbf{x}^2$ with $\mathbf{x}^3$ depending on $C_3 < C_{\max}$ or $C_3 > C_{\max}$, respectively. The algorithm starts with $\mathbf{x}^1$ as the quantizer output with minimum length, and $\mathbf{x}^2$ as the unconstrained quantizer output.

In the following, we prove some properties of the proposed interpolation approach.

Theorem 2: If $\mathbf{x}^1$ and $\mathbf{x}^2$ are the two interpolating points in the last iteration of the algorithm, then the interpolation between these two points results in one of them.

Proof: Assume the interpolation between $\mathbf{x}^1$ and $\mathbf{x}^2$ results in a new point, say $\mathbf{x}^3$, which is different from $\mathbf{x}^1$ and $\mathbf{x}^2$. This requires,

$$D_3 + \pi_1 C_3 < D_2 + \pi_1 C_2 = D_1 + \pi_1 C_1,$$

where by a direct substitution, we obtain,

$$D_3 + \pi_1 C_3 + \pi_2 < 0,$$

meaning that the convergence conditions is not satisfied and the inclusion of $\mathbf{x}^3$ results in a decrease in the objective function value. This contradicts the assumption that we are in final iteration.

Theorem 3: If $\mathbf{x}^1$ and $\mathbf{x}^2$ are the two interpolating points in the last iteration of the interpolation procedure, then $\mathbf{x}^1$ and $\mathbf{x}^2$ differ in at most one co-ordinate.

Proof: Proof follows using basic principles of linear programming as reflected in Theorem 1. According to Theorem 1, the algorithm ends with either all the components of the final solution being zero-one, or it will have two non-zero components along a single co-ordinate which should add to one. This second case can be interpreted as having an interpolation between two integer solutions, one satisfying the cost constraint and the other one violating it. This means that the two points involved in the interpolation procedure differ at most along one co-ordinate.

2.1 Linear Programming Approach to Fixed-rate Entropy-constrained Trellis Quantization

Trellis Coded Quantization (TCQ) relies on Ungerboeck's set partitioning ideas from trellis coded modulation. It uses an expanded signal set and set partitioning, where there is an assignment of partitions to trellis states such that the codewords in the partition are associated with edges emerging from the state. In a popular configuration of TCQ with a rate of R bits/sample, a scalar quantizer with 2^{R+1} elements is partitioned into four subsets (each with 2^{R-1} elements) and these subsets are then used to label the branches emerging from different states in a suitably chosen trellis [13].

Consider an N-dimensional TCQ (N as a block length) with $\nu = 2^{\mu}$ states. The corresponding scalar quantizer is specified by an alphabet (set of quantization levels) $Q = \{q_1, q_2, \cdots, q_{2M}\}$ where $M = 2^m$. The quantizer points are partitioned into 4 subsets, S_l, $l = 1, 2, \cdots, 4$, where each subset consists of $M/2$ points [13]. Assume that there is a variable-length, binary, prefix code associated with each subset[5]. The proposed entropy-constrained trellis-coded quantizer uses a proper Lagrange multiplier to adjoin the distortion to the code-word length and searches the trellis (using Viterbi algorithm) for a path to minimize this quantity. The branch metric for the Viterbi decoder is the sum of two terms, one term is the squared error between the current input and its closest codeword in the subset associated with that branch, and the other term is the Lagrange multiplier times the length of the corresponding codeword.

Similar to the previous case, the algorithm finds a chain of solutions each expressed in terms of the linear interpolation between two intermediate points. Starting with two points, say $\mathbf{x}^1$, and $\mathbf{x}^2$, we try to improve the solution by bringing a third point, say $\mathbf{x}^3$, into the interpolation procedure. By following a procedure similar to the case discussed earlier in (1) to (7), we reach to an equation similar to (8). The updating is achieved by solving a new LP problem for the selection of $\mathbf{x^3}$ which is subject to the trellis and indicator constraints, i.e.,

$$\begin{aligned} &\text{Minimize} \quad D_3 + \pi_1 C_3 = \sum_{i=1}^{N} \sum_{j=1}^{M} \left[d_i(j) + \pi_1 c(j) \right] x_i^3(j) \\ &\text{Subject to} \quad \text{Indicator and trellis constraints} \end{aligned} \tag{9}$$

Noting that the indicator constraints in (9) are decomposable, we conclude that the points along different dimensions can be selected independently (as far as the restrictions imposed by the structure of the underlying trellis is satisfied). This means that one can use the Viterbi algorithm to find the solution for (9) where the branch metric is a linear combination of distortion and code-word length of the form $d_i(j) + \pi_1 c(j)$.

In summary, given two points $\mathbf{x}^1$ and $\mathbf{x}^2$, the algorithm computes the value of π_1 and using the Viterbi algorithm finds the solution of (9). Then, one of the points $\mathbf{x}^1$ or $\mathbf{x}^2$ is updated and the iteration continues.

[5]These variable-length codes can also be designed for $S_1 \cup S_3$ and $S_2 \cup S_4$ [17].

3 Numerical Results

In the following, we present some numerical results for the performance and complexity of the proposed method for an i.i.d. Gaussian source. A Huffman code is used to label the quantizer points in all the cases. The Huffman code is designed to be optimum (minimum length) for the probabilities obtained in the last iteration of the employed iterative (LBG type) design algorithm. In all cases, a sequence of 20000 source vectors is used to design the quantizer and a different sequence of the same length is used to measure the resulting performance. The quantization is measured in terms of the mean square distance. The memory size is in kilo-bytes (8 bits) per N dimensions and the computational complexity is the number of floating points operations per dimension. In all cases, the complexity values given correspond to the "nearest neighbor search algorithm" (in our case, the labeling algorithm has a trivial complexity).

Table 1 gives an indication of the range of I (number of iterations) obtained experimentally from the simulation performed on our test data. In the following results, we have imposed the constraint $I \leq 10$ and the complexity values are given for $I = 10$. Note that the complexity values given correspond to the peak values and the average complexity would be lower.

		Entropy Coding		Huffman Coding	
N	M	$I_{\max}$	I_{avg}	$I_{\max}$	I_{avg}
32	8	9	1.89	9	3.27
64	8	9	2.99	9	4.69
128	8	11	4.54	11	6.33
128	16	11	5.02	11	7.10
256	8	11	6.27	11	8.04
256	16	11	7.82	11	8.48
512	8	13	8.17	12	9.55
512	16	13	9.00	12	9.57

Table 1: Number of iterations (1D case)

Table 2 shows the numerical results of the proposed quantizer for a rate of 2.5 bits/sample and for different dimensions. The first column is based on a constraint on the total Self Information (SI). By applying Huffman code and imposing the constraint on length instead of

self information, the performance drops about by 0.9 dB (HC-1D). This degradation will be generally lower for higher values of rate per dimension and can be reduced by merging two or more dimensions and applying a Huffman code to the resulting subspace. Column HC-2D in Table 2 corresponds to the case where two coordinates are merged.

Rate=2.5 bits/dimension, $M = 8$				
N	SI	HC-1D	HC-2D	$D(r)$-1.53†
32	13.02	11.91	12.51	13.53
64	13.20	12.25	12.83	13.53
128	13.29	12.45	13.03	13.53
256	13.36	12.57	13.24	13.53
512	13.39	12.64	13.30	13.53

Table 2: SNR (in dB) vs. dimension of the proposed method for a Gaussian source, 1D and 2D refer to using the Huffman code over 1 and 2 dimensional subspaces, respectively. † $D(r) - 1.53$ is the distortion rate function (in dB) of a Gaussian source, $D(r) = 10\log 10(2^{2r})$ minus 1.53dB where 1.53dB is the maximum possible quantization gain due to packing [23]. The subtraction of 1.53dB accounts for the lack of packing.

Table 3 provides the results when the proposed method is combined with a TCQ of 4 states and the underlying scalar quantizers are composed of 16 points (rate is 2.5 bits per sample). Column (HC-1D-TCQ) corresponds to the case that the Huffman codes are applied to the one-dimensional codewords. The results show more than 1 dB improvement in comparison with HC-1D given in Table 2 (gain of TCQ). Column HC-2D-TCQ corresponds to the case where the Huffman code is applied over two dimensions. The results show about 0.3 dB improvement in comparison with HC-1D-TCQ.

In Table 4, we have a comparison between the proposed method and the method presented in [10] and [11]. The new approach shows a substantial reduction in the complexity for a similar performance (at the price of a higher delay).

In Table 5, we have a comparison between the proposed method and the Lagrange-multiplier-based method presented in [2]. This method is mentioned in [2] to offer the best performance among the sub-optimum search methods examined. As mentioned earlier, in [2], the authors

Rate=2.5 bits/dimension $M = 8$			
N	HC-1D-TCQ	HC-2D-TCQ	$D(r)^\dagger$
32	13.34	13.63	15.05
64	13.56	13.93	15.05
128	13.73	14.13	15.05
256	13.80	14.25	15.05
512	13.84	14.32	15.05

Table 3: SNR (in dB) vs. dimension of the proposed method using a four state trellis (TCQ) for a Gaussian source, 1D and 2D refer to using the Huffman code over 1 and 2 dimensional subspaces, respectively. $^\dagger D(r)$ is the distortion rate function (in dB) of a Gaussian source.

Method	N	M	Rate	Computation	Memory (bytes)	SNR
HC-2D	128	8	2.5	200	0.4 k	13.03
[10]	16	8	2.5	2000	21 k	13.00
[11]	16	8	2.5	3000	0.22 k	12.91
HC-1D	128	16	3.5	380	0.6 k	18.8
[10]	32	16	3.5	10000	300 k	18.8
[11]	32	16	3.5	1100	21 k	18.7

Table 4: Comparison of proposed methods and references [10], [11]

increase the Lagrangian multiplier gradually from zero until the cost constraint is satisfied. They also mention the possibility of using a bisection method on λ for this purpose, but do not present any numerical results (we have implemented such a bisection method on λ for the sake of comparison).

Table 5 provides a comparison of the three approaches for updating λ where the complexity is expressed in terms of the number of iterations. In each iteration, all the three approaches have the same computational complexity (add/comparison). Therefore, in complexity comparisons in Table 5, we have just mentioned the number of iterations. For the proposed method, the two initial points are selected as the nearest quantizer output with the minimum length and the nearest quantizer output without considering the cost constraint. For the bisection method, the

initial Lagrangian multipliers are $\lambda = 0$ (corresponding to the nearest quantizer point without the cost constraint) and a λ large enough to guarantee a point with minimum total length (this is obtained experimentally from a large data set as the smallest possible value to guarantee a point with minimum overall length). Numerical results show the same performance at $R = 2.5$ bits/dimension and dimension $N = 32, 64$ for all the three methods, while the proposed method shows a substantial reduction in the complexity (number of iterations) as given in Table 5.

Method	N	M	Rate	Iteration	SNR
HC-1D	32	8	2.5	3.27	11.91
Bi-section	32	8	2.5	10.02	11.91
[2]	32	8	2.5	18.11	11.91
HC-1D	64	8	2.5	4.69	12.25
Bi-section	64	8	2.5	12.36	12.25
[2]	64	8	2.5	18.23	12.25

Table 5: Comparison of proposed method, bi-section and reference [2]

In Table 6, we have a comparison between proposed method and the method presented in [18]. As Table 6 shows, the new approach shows a substantial reduction in the complexity (with the same delay).

Rate=2.5 bits/dimension	$M = 8$, $N = 64$		
Method	Computations	Memory	SNR
HC-1D-TCQ	400	1.75 k	13.56
[18]	5720	15.5 k	13.56

Table 6: Comparison of proposed methods and trellis-based scalar-vector quantizer from reference [18]. The number of trellis states for both methods is four.

We have not provided a detailed comparison with the method of [1] because the results presented in [1] are based on a fine resolution approximation of the self information values requiring a high complexity for the underlying enumerative addressing scheme (the issue of addressing is not discussed in [1]). Note that unlike the method proposed here, the method

in [1] cannot be combined with a lattice quantizer either. As a rough comparison, in [1], a case is reported for $N = 32$ (refer to Table 3 of [1]) which results in an SNR of 13.0 dB for a rate of 2.5 bits/dimension where the number of operations for the nearest neighbor search is about 1700 per dimension and the memory requirement (RAM) is about 8K (this does not include the addressing complexity). These values can be compared with a case of 2.5 bit/dimension, $N = 128$, given in Table 4 where the computation complexity is 200 per dimension and memory requirement is 0.4 k (this includes addressing complexity).

We have not provided a detailed comparison with the method of [12] because it proposes a very low complexity search method which is proved to be optimum under some restrictive conditions on the code-word lengths and quantization distortion profile. In addition, unlike the method proposed here, the method discussed in [12] cannot be easily combined with trellis coding.

Acknowledgment: The help provided by Mr. Erik Hons in the computer simulation of part of the results provided in this paper is gratefully acknowledged.

References

[1] A. K. Khandani, "A Linear (Zero-one) Programming Approach to Fixed-rate, Entropy-Coded Vector Quantization," *IEE Proceeding Communication,* vol. 146, pp. 275–282, October 1999.

[2] A. S. Balamesh and D. L. Neuhoff, "Block-constrained methods of fixed-rate, entropy-Coded scalar quantization," *Technical report TR-350, University of Michigan,* pp. 1–60, Sept. 1992.

[3] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory,* Vol. IT-6, pp. 7-12, March 1960.

[4] S. P. Lloyd, "Least square quantization in PCM," *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 129–137, Jan. 1982.

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

[6] R. Blahut, "Computation of channel capacity and rate distortion functions," *IEEE Trans. Inform. Theory,* vol. IT-18, pp. 460–473, March 1972.

[7] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust., Speech, Signal Process.,* vol. 37, pp. 31–42, Jan. 1989.

[8] T. R. Fischer, "A pyramid vector quantizer," *IEEE Trans. Inform. Theory,* vol. IT-32, pp. 468–583, Nov. 1986.

[9] 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.

[10] 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.

[11] A. K. Khandani, "A Hierarchical Dynamic Programming Approach to Fixed-rate, Entropy-Coded Quantization," *IEEE Trans. Inform. Theory,* vol. IT-42, pp. 1298–1303, July 1996.

[12] S. Nikneshan and A. K. Khandani, "Sequential search approach to fixed-rate entropy-coded quantization," *Electronics Letters*, vol. 38, pp. 1262-1264, Oct. 2002.

[13] M. W. Marcellin and T. R. Fischer, "Trellis-coded quantization of memoryless and Gauss-Markov sources," *IEEE Trans. Commun.* vol. 38, pp. 82–93, Nov. 1990.

[14] T. R. Fischer, M. W. Marcellin, and M. Wang, "Trellis-coded vector quantization," *IEEE Trans. Inform. Theory,* vol. IT-37, pp. 1551–1566, Nov. 1991.

[15] H. S. Wang and N. Moayeri, "Trellis coded vector quantization," *IEEE Trans. Communication,* vol. 40, pp. 1273–1276, Aug. 1992.

[16] T. R. Fischer and M. Wang, "Entropy-constrained trellis-coded quantization," *IEEE Trans. Inform. Theory,* vol. IT-38, pp. 415–426, March 1992.

[17] M. W. Marcellin, "On entropy constrained TCQ," *IEEE Trans. Commun.,* vol. 42, pp. 14–16, Jan. 1994.

[18] R. Laroia and N. Farvardin, "Trellis-based scalar-vector quantizer for memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-40, pp. 860–870, May 1994.

[19] L. Yang and T. R. Fischer, "A new trellis source code for memoryless sources," *IEEE Trans. Inform. Theory,* vol. 44, pp. 3056–3063, Nov. 1998.

[20] H. M. Salkin, *Integer programming,* Addison-Wesley Publishing Company, 1975.

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

[22] George B. Dantzig "Linear Programming & Extensions," Princeton University Press, January 1998.

[23] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups,* Springer-Verlag, 1988.