Block-Constrained Methods of Fixed-Rate, Entropy-Coded, Scalar Quantization*

Ahmed S. Balamesh and David L. Neuhoff
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109

September 24, 1992

Abstract

Motivated by the recent work of Laroia and Farvardin, this paper presents new reduced-complexity methods for avoiding the buffering problems associated with entropy-coded, scalar quantization. Basically, given a fixed-size source block, these methods use dynamic programming and other techniques to search the sequences produceable by an entropy-coded, scalar quantizer for one with minimum distortion subject to a constraint on the number bits produced by some binary encoding of these sequences. The result is that although some encoding methods might have a variable rate on the sample level, the overall quantizer has a fixed rate on the block level. A general class of such methods, called *block-constrained quantization,* is introduced. A way to reduce the encoding complexity and several ways to to simplify the search complexity are found. A *node-varying* method with improved performance is given. New insight into the performance of block-constrained quantizers is presented. Compared to the original Laroia-Farvardin method, the results presented here show small improvements in performance and large reductions in complexity.

Key Words : Entropy Coding, Scalar Quantization, Structured Vector Quantization, Block-Constrained Quantization, Fixed-Rate Quantization.

1 Introduction

Quantization plays an important role in the transmission and storage of analog data.

It becomes increasingly important as demand presses against the capacity of available

*This work was supported in part by a scholarship from King Abdul-Aziz University, Jeddah, Saudi Arabia and by NSF grant NCR-9105647.

channels and storage media. Scalar quantization is the oldest form of quantization and the simplest. But, it is possible to do significantly better. For example, entropy coding improves the performance of scalar quantizers and can achieve performance within 1.53 dB of the rate-distortion limit, with respect to mean-squared error distortion.

The major disadvantage of entropy-coded quantizers is their variable output rate, which requires buffering and causes synchronization problems. Buffer-instrumented strategies have been used [1] to minimize such problems. However, such schemes degrade the performance of the system for future samples of the source if the past ones were bad. This means that the wrong samples get the "punishment".

A new fixed-rate method that resembles entropy-coded, scalar quantization has recently been proposed by Laroia and Farvardin [2]. With this approach, given a block of source symbols, dynamic programming is used to find a sequence of quantization levels that can be encoded with a fixed number of bits, using a kind of lexicographic indexing. Consequently, the source samples are not quantized independently and the samples which cause the rate to increase are the ones which are quantized with poorer fidelity.

Although this appears to be the right approach for achieving the performance of entropy-coded quantizers with fixed rate, there remain problems. Search complexity (per sample) is high and increases linearly with dimension, and the lexicographic indexing requires a large amount of storage.

In this paper, we suggest methods to substantially simplify the indexing and reduce the search complexity. To simplify the indexing, we make use of the fact that one can obtain a system that is fixed rate on a block level, while retaining the variable-rate nature at the sample level. To simplify the searching we show that with high

probability only a small, easily identified subset of the states need be searched by the dynamic programming. Moreover, we show that the dynamic programming can be replaced with far simpler, suboptimal methods, with only small losses in performance.

As additional motivation for work in this area, we cite, as did Laroia and Farvardin, the need for low-complexity, fixed-rate quantization at moderate-to-high rates. Although low rate (along with low distortion) is a primary goal of quantization, there are many systems where after suitable preprocessing (e.g. a transform or subband decomposition) there are important components of the data (e.g. low-frequency components) that must be quantized with very low distortion and, consequently, with moderate-to-high rate, e.g. three or more bits per sample. One might look to vector quantization (VQ), because VQ achieves the distortion-rate function (asymptotically as the dimension grows. However, at moderate-to-high rates, only small dimensions are practical, which greatly limits VQ performance. Even relatively low-complexity structured VQ's (such as two-stage and tree-structured) have large complexities in this range of rates. Thus, there is considerable need for low-complexity methods. Block-constrained methods such as those presented in [2] and here have complexity that increases slowly with rate. Thus, they are well suited to coding in the moderate-to-high rate range.

This paper is oriented towards achieving the performance of scalar quantization with first-order entropy coding, but with fixed-rate, low-complexity schemes. For memoryless sources and moderate-to-high rates, such performance is better than that achievable by practical VQ's. For sources with memory, VQ may work better, but it is believed that th methods described here can be extended to exploit source correlation, as for example in [3]. In this case, future work may find these methods to be

competitive with the best fixed-rate quantization techniques.

Throughout this paper, we use the squared-error distortion measure and results are given only for independent, identically-distributed (IID) sources, Gaussian and Laplacian; however, it is anticipated that the results extend to sources with memory, as for example in [3].

Section 2 gives background material on entropy-coded, scalar quantizers (ECSQ's). Section 3 introduces the problem of converting ECSQ's to fixed-rate. Although the first work in this area is that of Laroia and Farvardin, we describe our approach first, as it is more elementary. The more sophisticated Laroia-Farvardin approach is postponed to Section 4, where it will be seen to overcome some shortcomings of our approach at the expense of increased complexity. Section 4, presents our approach and that of Laroia and Farvardin [2] in a more unified framework. We use the term *block-constrained, fixed-rate, entropy-coded quantization* (or *block-constrained quantization* (BCQ) for short) to describe such schemes. Section 5 describes the implementation of the dynamic programming search. Section 6 discusses design and optimization of BCQ's. Section 7 compares the performance of the new and old BCQ methods. Section 8 gives the reduced state implementation of the dynamic programming search. Two very low-complexity search methods are presented in Sections 9 and 10. In Section 11, we present a version of BCQ, called node-varying BCQ, that reduces some of the shortcomings of method of Section 3. Finally, conclusions are given in Section 12.

2 Entropy-Coded Scalar Quantization

An entropy-coded, scalar quantizer (ECSQ) consists of an m-level scalar quantizer with levels $\underline{q} = (q_1, q_2, \ldots, q_m)$, where $q_1 < q_2 < \cdots < q_m$, thresholds $\underline{t} = (t_1, t_2, \ldots, t_{m-1})$, where $t_1 \leq t_2 \leq \cdots \leq t_m$, and a variable-length, binary, prefix code $C = \{c_1, c_2, \ldots, c_m\}$, where the binary codeword c_j corresponds to level q_j and has length l_j. Define $l(q_j) = l_j$, $l_{\min} = \min_j l_j$ and $l_{\max} = \max_j l_j$. Let $\underline{l} = (l_1, l_2, \ldots, l_m)$ denote the lengths of the binary codewords. It is well-known that there exists a prefix code with lengths $\underline{l}$ if, and only if, $\underline{l}$ has positive, integer components that satisfy Kraft's inequality, $\sum_{j=1}^{m} 2^{-l_j} \leq 1$. With this in mind, will always use $\underline{l}$ to describe a prefix code instead of C.

Define the quantization rule $Q_{\underline{t}}$ by $Q_{\underline{t}}(x) = q_j$ if $x \in (t_{j-1}, t_j]$, $j = 1, 2, \ldots, m$, where $t_0 \stackrel{\triangle}{=} -\infty$ and $t_m \stackrel{\triangle}{=} +\infty$. When there is no ambiguity, we will use Q and $Q_{\underline{t}}$ interchangeably. Finally, define the encoding rule $f_{\rm e}$ by $f_{\rm e}(q_j) = c_j$. Define $\bar{Q}$ to be the nearest-neighbor, quantization rule, i.e. $\bar{Q}(x) = q_j$ if $(x - q_j)^2$ is smallest.

Given a source sample x_i, the entropy-coded, scalar quantizer produces $Q_{\underline{t}}(x_i)$ and then transmits the corresponding binary codeword $f_e(Q_{\underline{t}}(x_i))$. We denote such a quantizer by $(\underline{q}, \underline{t}, \underline{l})$.

Let the source be modeled by an IID, discrete-time, random process $\{X_i : i = \ldots, -1, 0, 1, \ldots\}$ with zero mean and variance σ^2. The squared-error distortion, average length (rate) and entropy of a scalar quantizer $(\underline{q}, \underline{t}, \underline{l})$ are, respectively, $D_{\rm sq}(\underline{q}, \underline{t}) \stackrel{\triangle}{=} {\rm E}(X_i - Q_{\underline{t}}(X_i))^2$, $\bar{l}(\underline{t}, \underline{l}) \stackrel{\triangle}{=} {\rm E}l(Q_{\underline{t}}(X_i))$ and $H(\underline{t}) \stackrel{\triangle}{=} -\sum_{j=1}^{m} p_j \log_2 p_j$, where $p_j = \Pr(X_i \in (t_{j-1}, t_j])$ and $0 \log_2 0$ is set to 0.

We define the optimum performance theoretically achievable (OPTA) for entropy-

coded, scalar quantizers as

$$\delta_{\rm sq}^{\rm pe}(r) \stackrel{\triangle}{=} \inf_{\substack{m,\underline{q},\underline{t},\underline{l}: \\ \underline{l}\in\mathcal{L}_m^{\rm pe},\, \bar{l}(\underline{t},\underline{l})\le r}} D_{\rm sq}(\underline{q},\underline{t}), \tag{1}$$

where

$$\mathcal{L}_m^{\rm pe} = \left\{ (l_1, l_2, \ldots, l_m) : l_j \in \{1,2,\ldots\}, \sum_{j=1}^{m} 2^{-l_j} = 1 \right\},$$

and the superscript "pe" in $\delta_{\rm sq}^{\rm pe}$ and $\mathcal{L}_m^{\rm pe}$ is included to indicate that (sample-by-sample) prefix encoding is used. For a fixed $(\underline{q},\underline{l})$, we define the OPTA

$$\delta_{\rm sq}(\underline{q},\underline{l},r) \stackrel{\triangle}{=} \inf_{\underline{t}:\bar{l}(\underline{t},\underline{l})\le r} D_{\rm sq}(\underline{q},\underline{t}), \tag{2}$$

which characterizes the optimum performance of the ECSQ $(\underline{q},\underline{t},\underline{l})$ over all choices of thresholds $\underline{t}$.

The usual way of designing an entropy-coded, scalar quantizer is to first design a scalar quantizer with minimum distortion subject to a constraint on its output entropy, and then to use Huffman's algorithm to design a prefix code for the resulting quantizer. The rate of the prefix code (and the resulting ECSQ) equals the quantizer entropy plus a number between 0 and 1, called the redundancy of the prefix code.

Algorithms for designing quantizers with minimum distortion subject to a constraint on their entropy have been proposed in [6, 7, 8, 10, 11]. Such quantizers are called entropy-constrained, scalar quantizers, to distinguish them from the abovementioned entropy-coded, scalar quantizers[1]. The optimum performance of such quantizers is characterized by the following OPTA:

$$\delta_{\rm sq}^{\rm ent}(r) \stackrel{\triangle}{=} \inf_{m,\underline{q},\underline{t}:H(\underline{t})\le r} D_{\rm sq}(\underline{q},\underline{t}).$$

[1]In this paper, ECSQ will always mean entropy-coded, scalar quantization.

The results of the above papers numerically demonstrate that $\delta_{\rm sq}^{\rm ent}(r)$ is within about 1.5 dB of the distortion-rate function $\mathcal{D}(r)$, for a broad class of source distributions and all values of r. Moreover, it is known that uniform quantizers are nearly optimum for entropy-constrained quantization. The first discovery was that of Goblick and Holsinger [4] who numerically demonstrated that, for large values of r, uniform quantizers with output entropy r have distortion about 1.5 dB larger than $\mathcal{D}(r)$. Subsequently, Gish and Pierce [5] proved the optimality of uniform scalar quantizers for sufficiently smooth source densities and asymptotically large rates r. They also proved that $\lim_{r\to\infty}(\delta_{\rm sq}^{\rm ent}(r)/\mathcal{D}(r)) = \pi e/6$ (i.e. an increase of 1.53 dB). Wood [6] proved similar results. Moreover, it was demonstrated by several authors that even for moderate rates uniform quantizers are generally nearly optimum (see [6, 7, 9, 10]).

Figure 1 shows $\delta_{\rm sq}^{\rm ent}(r)$ expressed as signal-to-noise ratio vs. entropy, computed via Algorithm 2 of [10], for a Gaussian source. It also shows the performance of entropy-coded, scalar quantizers found by using Huffman's algorithm to generate prefix codes for the scalar quantizers that achieve $\delta_{\rm sq}^{\rm ent}(r)$. Note the 6 dB/bit slope of $\delta_{\rm sq}^{\rm ent}(r)$ and $\mathcal{D}(r)$ which is, in general, the case for large rates.

Using Huffman's algorithm to design prefix codes for optimum entropy-constrained, scalar quantizers does not necessarily result in the best entropy-coded, scalar quantizers. A better method for designing ECSQ's is motivated by the work of Chou, *et al.,* who proposed a training-sequence algorithm to design entropy-constrained, or entropy-coded, vector quantizers. In the case of scalar quantizers, it is straightforward to modify their algorithm to use the source density instead of a training sequence. The results of using this algorithm are also plotted in Figure 1. We make several observations. The redundancy for this algorithm is noticeably less

than for the previous approach, especially at rates 2 or below. Both ECSQ design approaches produce "scalloped" curves; i.e., the redundancy varies nonmonotonically with rate. Finally, the redundancy of each gets large as rate approaches one, which is due to the fact that prefix codes cannot have rate less than one.

3 Converting ECSQ to Fixed Rate

Consider the operation of the entropy-coded, scalar quantizer on a block basis. Given an n-dimensional source vector $\underline{x} = (x_1, x_2, \ldots, x_n)$, the entropy-coded, scalar quantizer finds $Q(\underline{x}) \triangleq (Q(x_1), Q(x_2), \ldots, Q(x_n))$ and transmits the binary string $f_e(\underline{x}) \triangleq f_e(Q(\underline{x})) \triangleq (f_e(Q(x_1)), f_e(Q(x_2)), \ldots, f_e(Q(x_n)))$, which is uniquely decodable due to the prefix property of C. The total length of the output string is $l(\underline{x}) \triangleq l(Q(\underline{x})) \triangleq \sum_i l(Q(x_i))$, which varies greatly with $\underline{x}$ and necessitates buffering, if transmission over a fixed-rate channel is intended.

A Naive Approach

We, first, consider a naive way to convert an entropy-coded, scalar quantizer $(\underline{q}, \underline{t}, \underline{l})$ into a fixed-rate, block code. Suppose we have a desired input blocklength n and rate r. Let us imagine having a buffer of length nr. Now given a source vector $\underline{x} = (x_1, x_2, \ldots, x_n)$, we successively apply the ECSQ to each source sample, placing the resulting binary codewords into the buffer. Once it is filled to capacity the nr bits are transmitted. Any further bits produced by the ECSQ are simply discarded. And if less than nr bits are produced, the buffer is filled to capacity with zeros. The decoder, which receives the nr bits, decodes them using the usual prefix decoder. If it decodes fewer than n samples, the remaining samples are reproduced with the mean value of the source. Although this system is very naive, it is easy to see that the

law of large numbers implies that if $r > \bar{l}(\underline{t}, \underline{l})$, then as n gets large, its distortion approaches $D_{\rm sq}(\underline{q}, \underline{t})$, the distortion of the ECSQ. Indeed, we show in Appendix A that this happens even if all samples in the block were to be reproduced by the mean value, when the decoder decodes fewer than n samples.

A Better Approach

A much better way to convert the ECSQ into a fixed-rate block code is the following: Given a source vector $\underline{x} = (x_1, x_2, \ldots, x_n)$, search for the closest sequence of n quantization levels, whose corresponding binary codewords have cumulative length at most nr. Clearly, such a system will give no larger distortion than the naive system. Hence, if n is large and $r > \bar{l}(\underline{t}, \underline{l})$, the distortion of the improved system will be no larger than $D_{\rm sq}(\underline{q}, \underline{t})$, the distortion of the ECSQ.

This method is actually just one of a family of methods that we call *block-constrained, fixed-rate, entropy-coded quantization* (or *block-constrained quantization* (BCQ) for short) that were inspired by the original pioneering work of Laroia and Farvardin [2]. We now give the details of the specific block-constrained method described above.

Given a source n-vector $\underline{x}$, the block-constrained quantizer finds $\underline{y} \in \{q_1, q_2, \ldots, q_m\}^n$ which solves the following minimization:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n} (x_i - y_i)^2, \\ \text{subject to} \quad & \sum_{i=1}^{n} l(y_i) \le L, \end{aligned} \tag{3}$$

where L is called the *length threshold* or simply the threshold. It should be noted that the above minimization has a solution if, and only if, $nl_{\rm min} \le L$, which will be assumed to hold throughout this paper and will not be mentioned later.

Let $\underline{\hat{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$ be a solution to the above problem. Then, the encoder transmits an $\lfloor L \rfloor$-bit string which is composed of the concatenation of the binary codewords corresponding to $\hat{y}_1$ through $\hat{y}_n$ plus $\lfloor L \rfloor - l(\underline{\hat{y}})$ zeros. This binary string is uniquely-decodable. Therefore, it is clear that the rate of this scheme is $\lfloor L \rfloor / n$ bits per source sample. Thus, on the block level, we have a fixed-rate quantizer, while retaining the inherent variable-rate nature on the sample level.

We call this kind of coding *prefix-encoded* block constrained quantization (pe-BCQ). We use the notation pe-BCQ$(\underline{q}, \underline{l}, n, L)$ to denote a specific code, whose distortion and rate are denoted $D_{\rm bcq}(\underline{q}, \underline{l}, n, L)$ and $R^{\rm pe}_{\rm bcq}(n, L) \stackrel{\triangle}{=} \lfloor L \rfloor / n$, respectively. This notation is intended to emphasize that the distortion does not depend on the prefix encoding scheme and the rate does not depend on the levels or the lengths.

The set of all sequences of n quantization levels whose cumulative length is at most L forms the *codebook*

$$\mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L) \stackrel{\triangle}{=} \left\{ \underline{y} \in \{q_1, q_2, \ldots, q_m\}^n : \sum_{i=1}^{n} l(y_i) \leq L \right\}. \tag{4}$$

Its members, called *codevectors*, comprise all possible solutions to (3).[2]

Finally, we define the OPTA for this class of BCQ's as:

$$\delta^{\rm pe}_{\rm bcq}(r) \stackrel{\triangle}{=} \inf_{n, m, \underline{q}, \underline{l}: \underline{l} \in \mathcal{L}^{\rm pe}_m} D_{\rm bcq}(\underline{q}, \underline{l}, n, nr),$$

which represents the best possible performance of prefix-encoded BCQ's.

It is interesting to compare the output of the BCQ with that of the nearest-neighbor, scalar quantizer with the same levels operating on the same block of source samples. If the latter has a total length no larger than L, then both outputs coincide, i.e. a valid output for one is valid for the other and they both have the same distortion.

[2] We emphasize that in our terminology, "codeword" means a *binary* sequence and "codevector" means a sequence of *quantization levels*.

Otherwise, the BCQ tries to find a sequence of quantization levels that is closest to the source sequence and satisfies the length constraint. Nevertheless, the two outputs will ordinarily be close to each other. Figure 2 shows a typical situation.

As an example, Figure 3 shows the performance of pe-BCQ based on a Huffman-coded, uniform scalar quantizer with 20 levels spaced $\Delta = .54$ apart, for various blocklengths n and an IID, unit variance Gaussian source. One may observe a clear improvement of the method with n. The figure also shows the performance of the naive system based on the same levels and lengths, with nearest neighbor thresholds. Notice that the block-constrained approach does significantly better than the naive one.

Explaining the Asymptotic Behavior

Note that the above block-constrained method never actually uses the thresholds $\underline{t}$ of the underlying scalar quantizer. Indeed, it is important to observe that given any thresholds $\underline{t}'$, if one quantizes a source vector $\underline{x}$ with the scalar quantizer $(\underline{q}, \underline{t}', \underline{l})$ and obtains $l(\underline{x}) \leq nr$, then pe-BCQ$(\underline{q}, \underline{l}, n, nr)$ will find a $\underline{y}$ that is at least as good as that produced by the scalar quantizer. It follows from the law of large numbers that if $r > \bar{l}(\underline{t}', \underline{l})$ and n is large, then pe-BCQ$(\underline{q}, \underline{l}, n, nr)$ will have distortion no larger than that of the scalar quantizer $(\underline{q}, \underline{t}', \underline{l})$. That is, for any $\underline{t}'$ such that $\bar{l}(\underline{t}', \underline{l}) < r$,

$$\limsup_{n \to \infty} D_{\mathrm{bcq}}(\underline{q}, \underline{l}, n, nr) \leq D_{\mathrm{sq}}(\underline{q}, \underline{t}').$$

By taking the infimum of the above over all $\underline{t}'$ such that $\bar{l}(\underline{t}', \underline{l}) < r$, it follows that

$$\limsup_{n \to \infty} D_{\mathrm{bcq}}(\underline{q}, \underline{l}, n, nr) \leq \delta_{\mathrm{sq}}(\underline{q}, \underline{l}, r),$$

where $\delta_{\rm sq}(\underline{q},\underline{l},r)$ is defined by (2) and assumed to be continuous at r.[3] Indeed, it is shown in [16] that

$$\inf_n D_{\rm bcq}(\underline{q},\underline{l},n,nr) = \lim_{n\to\infty} D_{\rm bcq}(\underline{q},\underline{l},n,nr) = \delta_{\rm sq}(\underline{q},\underline{l},r), \tag{5}$$

assuming $\delta_{\rm sq}(\underline{q},\underline{l},r)$ is continuous at r. Moreover, it is clear that

$$\lim_{n\to\infty} R^{\rm pe}_{\rm bcq}(n,nr) = \lim_{n\to\infty} \frac{\lfloor nr \rfloor}{n} = r. \tag{6}$$

Therefore, the limiting distortion of pe-BCQ$(\underline{q},\underline{l},n,nr)$ is, precisely, the least distortion of entropy-coded scalar quantization with levels $\underline{q}$, lengths $\underline{l}$ and any thresholds $\underline{t}$ such that $\bar{l}(\underline{t},\underline{l}) \le r$.

It follows easily that

$$\delta^{\rm pe}_{\rm bcq}(r) = \delta^{\rm pe}_{\rm sq}(r), \tag{7}$$

assuming $\delta^{\rm pe}_{\rm sq}$ is continuous at r, where $\delta^{\rm pe}_{\rm sq}$ is as defined by (1). This means that the best performance achieved by the class of BCQ's introduced so far (i.e. prefix-encoded BCQ's) is, exactly, the best performance achieved by ECSQ's. (See [16] for more details.)

It is important to notice that (5) holds for any choice of $\underline{l}$; it does not require $\underline{l}$ to satisfy Kraft's inequality. Indeed, it holds for any real lengths (even negative ones). This makes the result a very useful tool for the asymptotic analysis of the generalized BCQ's discussed in Section 4, below.

Finally, with our new understanding that BCQ does not depend on the quantization thresholds, we reconsider the comparison with the naive system. In Figure 3, the naive system used nearest-neighbor quantization, i.e. used a fixed set of thresholds.

[3] This is expected for continuous sources and $r > l_{\min}$.

But, for a given rate r, it may be possible to choose a better set of thresholds. Thus, the comparison in Figure 3 is unfair to the naive system. To make a fairer comparison, we ran the naive system with a variety of threshold sets, and for each r we plot in Figure 4 the best signal-to-noise ratio attained. Motivated by the work of [11], for a range of $\lambda \geq 0$, we considered sets of thresholds that minimized the functional $(x - q_j)^2 + \lambda l_j$ over j. The same figure also shows the performance of BCQ as well as its asymptotic distortion limit $\delta_{\text{sq}}(\underline{q}, \underline{l}, r)$. It can be seen that the naive system improved considerably, but is still substantially, inferior to the BCQ.

Effects of Prefix Code Redundancy

As mentioned above, scalar quantization with first-order prefix encoding cannot achieve the OPTA of entropy-constrained, scalar quantization, due to code redundancy. A typical redundancy of .03 bits/sample (see Figure 1) translates into a loss of about .2 dB in signal-to-noise ratio (SNR). As shown above, this loss is inherited by any pe-BCQ. Of course, this redundancy can be reduced by applying a prefix code to blocks of quantizer levels. However, this will increase the complexity of the BCQ.

As we will explain later, Laroia and Farvardin [2] avoid this redundancy by assigning non-integer lengths to the quantization levels. This, however, means that the binary encoding/decoding can no longer be done on a sample-by-sample basis. We will discuss this in more detail in the next section.

4 General Block-Constrained Quantizers

In this section, we describe a more general class of block-constrained quantizers motivated by the case explained above and by the work of Laroia and Farvardin [2].

As so far introduced, a BCQ is characterized by quantization levels $\underline{q}$, lengths $\underline{l}$ (positive integers satisfying Kraft's inequality), a blocklength n and a threshold L. Its *codebook* consists of the set of quantization level sequences $\underline{y}$, called *codevectors*, whose cumulative length $l(\underline{y})$ does not exceed the threshold L. A given source sequence $\underline{x}$ is quantized into the closest codevector in the codebook (i.e. we find the closest sequence of quantization levels whose cumulative length is no larger than L). And each component of this codevector is encoded into the corresponding codeword from some binary prefix code with lengths $\underline{l}$.

A generalized BCQ is the same except that we allow the lengths in $\underline{l}$ to be arbitrary, real numbers (even negative) and we permit other methods to binary-encode the codevectors. As before, the codebook, denoted by $\mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L)$, is given by (4), where L may need to be different from nr, depending on the particular binary encoding scheme, i.e. L is chosen so that the binary encoding scheme has a rate equal to the desired rate r.

The quantization process proceeds as before via (3). If $\underline{l}$ consists of rational (or rationalizable) lengths with a relatively small denominator, then the quantization can be implemented efficiently using dynamic programming as proposed in [2] and explained in Section 5. If $\underline{l}$ consists of arbitrary real numbers, the implementation of (3) can be complex; however, low-complexity, approximations are possible. Such methods will be considered in Sections 9 and 10.

There are many ways to binary-encode the codevectors into bits. However, the minimum possible rate is achieved by fixed-length *block encoding*, whose rate is

$$R^{\rm be}_{\rm bcq}(\underline{l}, n, L) \stackrel{\triangle}{=} \frac{1}{n} \left\lceil \log_2 \left| \mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L) \right| \right\rceil$$

bits per source sample, where $\left| \mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L) \right|$ is the size of the BCQ codebook. The

Laroia and Farvardin scheme used block encoding.

Conversely, when it is desired to achieve rate r with block encoding, the threshold L should be chosen so that the codebook is as large as possible without exceeding $2^{\lfloor nr \rfloor}$. There may be several suitable values. We let $L^{\rm be}(\underline{l}, n, r)$ denote any such value. With this choice of threshold, we are guaranteed to have $R^{\rm be}_{\rm bcq}(\underline{l}, n, L^{\rm be}(\underline{l}, n, r)) \leq r$, and it can happen that the inequality is strict. However, it is shown in [16] that $\lim_{n\to\infty} R^{\rm be}_{\rm bcq}(\underline{l}, n, L^{\rm be}(\underline{l}, n, r)) = r$. Moreover, a constant $c_{\underline{l}}(r)$ is found such that $\lim_{n\to\infty} L^{\rm be}(\underline{l}, n, r)/n = c_{\underline{l}}(r)$, which shows the asymptotic behavior of $L^{\rm be}(\underline{l}, n, r)$ for large n. We will also need the result, shown in Appendix B, that if $\underline{l}$ satisfies Kraft's inequality and $L < nl_{\max}$, then[4]

$$\frac{1}{n}\log_2 \left|\mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L\right| \;<\; \frac{L}{n}. \tag{8}$$

This kind of block constrained quantization will be called *block encoded BCQ* (be-BCQ). Its OPTA is

$$\delta^{\rm be}_{\rm bcq}(r) \;\stackrel{\triangle}{=}\; \inf_{n,m,\underline{q},\underline{l},L:R^{\rm be}_{\rm bcq}(\underline{l},n,L)\leq r} D_{\rm bcq}(\underline{q}, \underline{l}, n, L).$$

Laroia and Farvardin [2] showed[5]

$$\delta^{\rm be}_{\rm bcq}(r) \;\leq\; \delta^{\rm ent}_{\rm sq}(r). \tag{9}$$

Therefore, be-BCQ is, asymptotically, at least as good as optimum, entropy-constrained scalar quantization. Moreover, for any fixed $\underline{q}$ and $\underline{l}$, they show that the limiting performance of be-BCQ cannot be better than entropy-constrained scalar quantization, i.e.

$$\liminf_{n\to\infty} D_{\rm bcq}(\underline{q}, \underline{l}, n, L^{\rm be}(\underline{l}, n, r)) \;\geq\; \delta^{\rm ent}_{\rm sq}(r). \tag{10}$$

[4]See [16] for a more general version that does not require Kraft's condition.

[5]Although they do not explicitly mention it, this, and the following, result require $\delta^{\rm ent}_{\rm sq}$ to be continuous at r.

They also make the plausible conjecture that, even for small values of n, be-BCQ cannot do better that $\delta_{\rm sq}^{\rm ent}(r)$. Assuming so, then $\delta_{\rm bcq}^{\rm be}(r) = \delta_{\rm sq}^{\rm ent}(r)$. One should compare this result to (7), which shows that the best performance of pe-BCQ is $\delta_{\rm sq}^{\rm pe}(r)$. Alternative, more constructive proofs for (9) and (10) are given in [16].

In view of the above, a be-BCQ based on an optimum, entropy-constrained scalar quantizer seems a good idea. Specifically, given r, let $\underline{q}$ and $\underline{t}$ achieve $\delta_{\rm sq}^{\rm ent}(r)$, i.e. $H(\underline{t}) \approx r$ and $D_{\rm sq}(\underline{q}, \underline{t}) \approx \delta_{\rm sq}^{\rm ent}(r)$ and let $l_j = -\log_2 p_j$, where p_j is the probability of the j-th level q_j. Then, $\bar{l}(\underline{t}, \underline{l}) = H(\underline{t}) \approx r$. Now consider[6] be-BCQ($\underline{q}, \underline{l}, n, nr$). From (5), we have, for large n,

$$D_{\rm bcq}(\underline{q}, \underline{l}, n, nr) \approx D_{\rm sq}(\underline{q}, \underline{t}) \approx \delta_{\rm sq}^{\rm ent}(r),$$

and using (8) and the fact that $\underline{l}$ (as chosen) satisfies Kraft's inequality, we have

$$R_{\rm bcq}^{\rm be}(\underline{l}, n, nr) \lesssim r,$$

So, this BCQ operates at approximately $\delta_{\rm sq}^{\rm ent}(r)$, i.e. it is a good BCQ. Accordingly, we consider the lengths $l_j = -\log_2 p_j$ to be *ideal lengths.*

We make a last remark. As demonstrated in Section 3, when $\underline{l}$ consists of positive integers satisfying Kraft's inequality, a BCQ based on $\underline{l}$ can be encoded, in a very simple manner, using a prefix code with lengths $\underline{l}$. (This is pe-BCQ.) In this case, L (an integer) becomes the number of produced bits and $r = L/n$. By (8), we can see that, $r \geq R_{\rm bcq}^{\rm be}(\underline{l}, n, L)$. Indeed, in most cases, $r > R_{\rm bcq}^{\rm be}(\underline{l}, n, L)$, which results in some loss in performance versus a system using block encoding of the BCQ codebook. This loss, which is an example of what we call a *code-space loss,* and other losses encountered in BCQ will be discussed later.

[6] For this case, as shown in [16], $c_{\underline{l}}(r) = 1$, so $L^{\rm be}(\underline{l}, n, r) \approx nr$ is a good choice.

Laroia-Farvardin Approach

As we have seen, the pe-BCQ's introduced in Section 3 inherited the redundancy loss of first-order prefix codes. Laroia and Farvardin's approach suffers less from this kind of loss because it uses non-integer lengths $\underline{l}$, which make it better able to approximate the ideal lengths of the form $-\log_2 p_j$. However, because of this choice, they cannot simply do sample-by-sample, binary encoding of the levels with a prefix code. Instead, they use the block encoding described earlier, which also reduces the code-space loss mentioned earlier.

To simplify the search in (3), they make the lengths rational with a relatively small denominator b (typically 4). Moreover, this enables the block encoding to make use of a lexicographic ordering, which is a substantial simplification. The larger b, the less the redundancy loss, but the more complex the search and the lexicographic encoding.

They used the threshold $L^{\mathrm{be}}(\underline{l}, n, r)$, introduced before, and reported it to be approximately equal to $1.5bnr$.

Henceforth, we will use lf-BCQ to refer to the Laroia-Farvardin quantizer. (It is be-BCQ with rational lengths and lexicographic encoding.)

Losses in BCQ

Since BCQ's asymptotically achieve the performance of entropy-constrained scalar quantization, it is interesting to explore the losses incurred by BCQ's due to practical considerations. Below, we give a brief account of such losses.

First, we note from the above that, in general, for BCQ's to achieve $\delta_{\mathrm{sq}}^{\mathrm{ent}}(r)$, ideal, real lengths must be used. Therefore, when, for practical reasons (such as simplifying

the implementation of (3) and/or the binary encoding), the lengths are restricted to a subset of the reals (e.g. integers in the case of pe-BCQ and rationals with denominator b in lf-BCQ), then some loss is incurred.

Another loss suffered by BCQ systems is that their rates are generally larger than $\left\{\log_2 \left|\mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L)\right|\right\}/n$, by an amount that we call a *code-space* loss. Generally speaking, the code-space loss is larger for BCQ's that use a suboptimal binary encoding scheme (e.g. the sample-by-sample, prefix binary encoding in pe-BCQ) than for block-encoded BCQ's, which have the the least possible code-space loss. Indeed their code-space loss goes to zero as n tends to infinity. For pe-BCQ, the sum of the two losses (length-restriction and code-space loss) corresponds to the redundancy of the prefix code.

Finally, there is a *finite-dimensional loss* when n is not large enough for the law of large numbers to entirely dictate performance.

Relationship to Permutation Codes

Permutation codes are fixed-rate, block codes that have been shown to achieve $\delta_{\rm sq}^{\rm ent}(r)$ [13, 7, 9]. Therefore, it is interesting and important to compare them to BCQ. Specifically, $\mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L)$ can be viewed as a union of permutation codebooks. To see this, we define the type of a codevector $\underline{y} \in \{q_1, q_2, \ldots, q_m\}^n$ as the m-tuple $\underline{n} = (n_1, n_2, \ldots, n_m)$ where n_j is the number of times q_j appears in $\underline{y}$. We denote the type of $\underline{y}$ by $\mathcal{T}(\underline{y})$. Let $\mathcal{C}_{\underline{q}}^n(\underline{n}) \stackrel{\triangle}{=} \{\underline{y} \in \{q_1, q_2, \ldots, q_m\}^n : \mathcal{T}(\underline{y}) = \underline{n}\}$. Given L and $\underline{l}$ define the set of admissible types $\mathcal{N}(\underline{l}, n, L) \stackrel{\triangle}{=} \{\underline{n} : n_j \geq 0,\ \text{integer},\ j = 1, 2, \ldots, m, \sum_{j=1}^m n_j = n, \sum_{j=1}^m n_j l_j \leq L\}$. Then, the block-constrained quantizer codebook is given by

$$\mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L) = \bigcup_{\underline{n} \in \mathcal{N}(\underline{l}, n, L)} \mathcal{C}_{\underline{q}}^n(\underline{n}).$$

Moreover, for any $\underline{n}$, $\mathcal{C}_{\underline{q}}^{n}(\underline{n})$ is a the codebook of a variant-I permutation code [13]. Thus, a BCQ codebook is the union of permutation codebooks. It is interesting to note that the lengths $\underline{l}$ and L determine which types are included and which are not. Thus, careful choice of $\underline{l}$ and L can be used to construct codes with special features.

Shannon theory indicates that for very large n, optimal codes can be restricted to the surface of an n-dimensional sphere. Permutation codes do this. However, if n is not sufficiently high, then it is better to distribute the codewords over a spherical shell whose thickness increases with rate. This explains the observation in [13] that for a given dimension n, the distortion of permutation codes diverges from $\delta_{\mathrm{sq}}^{\mathrm{ent}}(r)$ as r increases. BCQ's overcome this problem by mixing into the codebook multiple permutation codes lying on spheres of different radii.

5 Dynamic Programming Implementation of BCQ Search

The goal is to efficiently perform the minimization in (3) that defines the quantization rule of a block constrained quantizer BCQ($\underline{q}, \underline{l}, n, L$). First, consider the case where $\underline{l}$ consists of positive integers. In this case the minimization can be implemented using dynamic programming as in [2]. For completeness and for future reference, we describe it here. Assume that a source sequence $\underline{x}$ is given. Let $\underline{y}^*$ denote a minimizing sequence of levels and D^* denote the resulting distortion. The minimization in (3) is simplified by the introduction of the notion of state. Specifically, the "state" of a k-tuple of quantization levels $(y_1, y_2, \ldots, y_k) \in \{q_1, q_2, \ldots, q_m\}^k$ is defined to be

$$s_k = s(y_1, y_2, \ldots, y_k) \stackrel{\triangle}{=} \sum_{i=1}^{k} l(y_i),$$

and we take $s_0 = 0$. Let S_k denote the set of possible values for s_k. Since we are only interested in $\underline{y} = (y_1, y_2, \ldots, y_n)$ that satisfy the length constraint $\sum_i l(y_i) \leq L$, we remove from S_k any values larger than L. Thus, there are at most L states in S_k.

In dynamic programming one recursively solves for $\underline{y}^*$ and D^*. Specifically, for $k \in \{1, 2, \ldots, n\}$ and $s \in S_k$, let

$$D^*_{k,s} = \min_{(y_1,y_2,\ldots,y_k):s(y_1,y_2,\ldots,y_k)=s} \sum_{i=1}^{k} (x_i - y_i)^2,$$

and let $\underline{y}^*_{k,s}$ denote the k-tuple that achieves the minimum in the above. Given that one has already found $D^*_{k,s}$ and $\underline{y}^*_{k,s}$ for a given k and all $s \in S_k$, one recursively finds $D^*_{k+1,s}$ and $\underline{y}^*_{k+1,s}$ as follows:

$$\begin{aligned} D^*_{k+1,s} &= \min_{j \in \{1,2,\ldots,m\}} D^*_{k,s-l_j} + (x_{k+1} - q_j)^2, \ s \in S_{k+1}, \\ \underline{y}^*_{k+1,s} &= (\underline{y}^*_{k,s-l_j}, q_{\hat{\jmath}}), \ s \in S_{k+1}, \end{aligned}$$

where $\hat{\jmath}$ solves the above minimization and $D^*_{k,s} = \infty$, for any $s \leq 0$, except $D^*_{0,0} = 0$. Then

$$D^* = \min_{s \in \{1,\ldots,L\}} D^*_{n,s}$$

and

$$\underline{y}^* = \underline{y}^*_{n,\hat{s}},$$

where $\hat{s}$ solves the above minimization.

The above can be viewed as a search over a non-uniform trellis as illustrated in Figure 5. At depth k, the trellis has a state corresponding to every possible sum of k lengths, e.g. state s represents all the quantizer k-tuples whose total length is s. Thus, every state has m branches entering it, where the branch labeled with q_j (or

$(x_k - q_j)^2$) comes from the state $s - l_j$ at depth $k-1$. The maximum number of states at any depth is no larger than the maximum allowable total length L. Roughly speaking, this search requires storage proportional to nL (which is, approximately, the total number of states in the trellis) and computational effort proportional to mL arithmetic operations per source sample.

For rational lengths with denominator b, the search is performed on the equivalent codebook $\mathcal{C}_{\rm bcq}(\underline{q}, b\underline{l}, n, bL)$, so that bL replaces L in the algorithm and, hence, in the storage requirements and the computational effort.

6 Design and Optimization

Although for large n, as discussed earlier, the block-constrained quantizers can be based on the levels and lengths of an optimum entropy-coded/entropy-constrained, scalar quantizer, for moderate values of n, it helps to optimize the levels and lengths for the given n and r. To do so, one may alternately optimize the lengths $\underline{l}$ for given levels $\underline{q}$, and vice versa.

1. **Optimizing $\underline{q}$ given $\underline{l}, n$ and L**

   Given $\underline{l}$, one can optimize $\underline{q}$ using the method described in [2]. This is, actually, a generalized LBG [14, 15] algorithm for trellises, which guarantees monotonic decrease in distortion per every iteration.

   We describe this algorithm very briefly. Given a training sequence $\{\underline{x}^1, \underline{x}^2, \ldots, \underline{x}^K\}$ of n-dimensional source vectors. Let $\underline{y}^k$ be the output of BCQ($\underline{q}, \underline{l}, n, L$) corresponding to $\underline{x}^k$. The algorithm replaces $\underline{q}$ by $\tilde{\underline{q}}$ =

$(\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_m)$ where

$$\tilde{q}_j = \frac{1}{|S_j|} \sum_{(k,i)\in S_j} x_k^i,\ j = 1, 2, \ldots, m, \tag{11}$$

and $S_j = \{(k,i) : y_i^k = q_j, i = 1, \ldots, n, k = 1, \ldots, K\}$, i.e. q_j is replaced by the centroid of the set $\{x_i^k : (k,i) \in S_j\}$.

Now, let $\underline{\tilde{y}}^k$ be such that $\tilde{y}_i^k = \tilde{q}_j$ if $y_i^k = q_j$. Thus,

$$\begin{aligned}
\sum_{k=1}^{K} \|\underline{x}^k - \underline{\tilde{y}}^k\|^2 &= \sum_{k=1}^{K}\sum_{i=1}^{n}(x_i^k - \tilde{y}_i^k)^2 \\
&= \sum_{j=1}^{m}\sum_{(k,i)\in S_j}(x_i^k - \tilde{q}_j)^2 \\
&\le \sum_{j=1}^{m}\sum_{(k,i)\in S_j}(x_i^k - q_j)^2 \\
&= \sum_{k=1}^{K}\sum_{i=1}^{n}(x_i^k - y_i^k)^2, \\
&= \sum_{k=1}^{K}\|\underline{x}^k - \underline{y}^k\|^2.
\end{aligned}$$

Noting that $\underline{\tilde{y}}^k \in \mathcal{C}_{\rm bcq}(\underline{\tilde{q}}, \underline{l}, n, L)$, we can easily see that the distortion of the next iteration will not be larger than $\sum_k \|\underline{x}^k - \underline{\tilde{y}}^k\|^2/K$ and hence, the distortion will be non-increasing.

2. Optimizing $\underline{l}$ given $\underline{q}, n$ and L

As before, consider a training sequence $\{\underline{x}^k\}$ and let $\{\underline{y}^k\}$ be as above. Let $\hat{p}_j$ be the frequency with which the level q_j is used, i.e. $\hat{p}_j \stackrel{\triangle}{=} n_j/nK$, where

$$\begin{aligned}
n_j &\stackrel{\triangle}{=} |S_j| = \sum_{k=1}^{K} n_j^k, \\
n_j^k &\stackrel{\triangle}{=} \left|\{(k,i) : y_i^k = q_j, i = 1, 2, \ldots, n\}\right|.
\end{aligned}$$

For the moment, suppose that $K = 1$, i.e. we have only one training vector. Then, $(n_1, n_2, \ldots, n_m)$ is the type of $\underline{y}^1$ and $\sum_j n_j l_j \le L$. Now, we replace $\underline{l}$

by $\underline{\tilde{l}}$ such that $\sum_j n_j \tilde{l}_j$ is minimum over all possible $\underline{l}$ of interest (e.g. positive integers satisfying Kraft's inequality in the case of pe-BCQ, or positive rationals with denominator b in the case of lf-BCQ, etc.). Note that this is equivalent to minimizing $\sum_j \hat{p}_j l_j$. Thus, we have $\sum_i \tilde{l}(y_i^1) = \sum_j n_j \tilde{l}_j \leq \sum_j n_j l_j \leq L$ and, therefore, $\underline{y}^1 \in \mathcal{C}_{\rm bcq}(\underline{q}, \underline{\tilde{l}}, n, L)$. Consequently, in the next iteration the distortion can be at most $\|\underline{x}^1 - \underline{y}^1\|^2$. It is interesting to note that the problem of finding $\underline{\tilde{l}}$ is the same as finding lengths $\underline{\tilde{l}}$ that minimize the average length for a source that has probabilities $\{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_m\}$. Therefore, if $\{l_1, l_2 \ldots, l_m\} \subset \mathbb{R}^+$ then $\tilde{l}_j$ can be chosen as $-\log_2 \hat{p}_j$ and if $\underline{\tilde{l}}$ must contain positive integers satisfying Kraft's inequality, then $\underline{\tilde{l}}$ corresponds to the lengths of codewords produced by Huffman's procedure applied to a source with probabilities $\{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_m\}$, etc.

The above argument no longer holds if $K > 1$, since, in this case, minimizing $\sum_j n_j l_j = \sum_k \sum_j n_j^k l_j \leq KL$ does not guarantee that the individual sums $\sum_j n_j^k l_j$ are reduced. Thus, $\underline{y}^k$ is no longer guaranteed to belong to $\mathcal{C}_{\rm bcq}(\underline{q}, \underline{\tilde{l}}, n, L)$. Nevertheless, this algorithm can be tried. In many cases, it results in a reduction in distortion.

Another less intuitive algorithm has been proposed by Laroia and Farvardin in [2], which they argue is asymptotically optimum in n.

3. Choice of initial $\underline{q}$ and $\underline{l}$

Initially, $\underline{q}$ and $\underline{l}$ can be obtained using a good ECSQ design algorithm and a Huffman design for the lengths for the case of pe-BCQ. The method of Chou, *et al.,* [11] (which is a generalization of Algorithm 2 in [10]) is quite a good choice. We tried both the training-sequence and density versions of such. For high rates,

we found that a uniform scalar quantizer with a Huffman code is a sufficiently good initial choice. Also, the optimized $\underline{q}, \underline{l}$ found for high dimensions are good initial choices for smaller dimensions.

For BCQ's using other encoding schemes similar considerations may still hold, as in the case of lf-BCQ [2].

7 Comparison of pe-BCQ and lf-BCQ

Complexity

Table 1 gives rough complexity estimates for the dynamic programming implementation of the codebook search for pe-BCQ ($L = nr$) and lf-BCQ using rational lengths with denominator b ($L \approx 1.5bnr$.) Only the dominant terms are shown. Also, the table shows corresponding estimates for the complexity of the binary encoding. From the table, we see that, a pe-BCQ with dimension approximately $1.5bn$ has almost the same search complexity as lf-BCQ with dimension n. Moreover, pe-BCQ has a trivial binary encoding complexity as compared to lf-BCQ, particularly in storage requirements.

Performance

Table 2 shows the signal-to-noise ratios of pe-BCQ for rates $1.5, 2, 2.5, 3$ and dimensions $48, 96, 144, 192$, for IID Gaussian and Laplacian sources. The signal-to-noise ratio of lf-BCQ with dimension 32 and $b = 4$, as reported in [2], is also included, to compare with the pe-BCQ with dimension 192. In this case, pe-BCQ has the same search complexity as lf-BCQ, while the former has comparatively negligible encoding complexity and the latter requires six times less storage for the search. Overall, the pe-BCQ is less complex and requires less storage. Also, included in Table 2 are the

performances of entropy-constrained[7] and entropy-coded scalar quantizers (using the Chou, *et al.,* algorithm mentioned in Section 2).

According to (5)-(7), the performance of pe-BCQ should approach that of entropy-coded scalar quantization (ECSQ) as n gets large. We see from the table that with $n = 192$, the signal-to-noise ratio of pe-BCQ is less than that of ECSQ by about .1 to .6 dB. As explained earlier, entropy-constrained quantization has higher SNR than ECSQ, typically two or three tenths of a dB. (An exception is the Laplacian source at rate 1.5.)

The decrease in rate that would result by replacing the prefix encoding with block encoding is shown in parentheses in Table 2. Notice that these numbers decrease with dimension n, which corresponds to the fact that, generally speaking, less code-space loss is incurred by pe-BCQ at larger dimensions, so there is less to be recovered by block encoding. Multiplying these numbers by 6 gives, approximately, the gain in dB that would result from block encoding. Typically, it is about .1 dB.

For the Gaussian source the table shows that at rates 1.5 and 2, lf-BCQ with $n = 32$ is better than pe-BCQ with $n = 192$. However, at rates 2.5 and 3, pe-BCQ is slightly better than lf-BCQ. For the Laplacian source, the performance is similar but tilts in favor of pe-BCQ. For example, at rate 3 pe-BCQ is about .5 dB better. The table also shows that the performance of pe-BCQ degrades gracefully as n decreases.

The abovementioned behavior can be explained in terms of the previously mentioned BCQ losses. While pe-BCQ suffers more length-restriction loss than lf-BCQ, it suffers less finite-dimensional loss, because of the larger n at which it can operate with reasonable complexity. Moreover, its larger dimension reduces its code-space loss. The fact that pe-BCQ performs better, relative to lf-BCQ, on the Laplacian

[7] These are quoted from [2].

source can be understood by noting that the Laplacian density has heavier tails than the Gaussian (in a sense, it's more nearly uniform), so the codeword lengths are more uniform. This causes the variance of the length produced by a nearest neighbor quantizer to be smaller, which in turn causes the law of large numbers to activate at smaller dimensions, which makes the dimension advantage more pronounced for a Laplacian source than for a Gaussian. (Notice that the SNR increases more with n for the Laplacian than for the Gaussian.)

The degradation of the performance of pe-BCQ as the rate gets smaller can be understood in view of the fact that performance of pe-BCQ follows that of ECSQ, which, as shown in Figure 1, degrades as the rate approaches one. On the other hand, lf-BCQ is not so limited, and performs better than ECSQ at small rates.

Finally, we comment on the effect of m on the performance of BCQ's. One is tempted to believe that the performance of a BCQ must improve as m increases. However, the subtle behavior of the number of levels in entropy-constrained quantizers [12, 10], as well as entropy-coded quantizers, makes the issue less trivial. Moreover, the effects of m on the code-space loss is not at all clear. For these reasons, we feel that the effect of m needs to be extensively investigated and, thus, we did not try to investigate it here. We leave it for future research. The reader is referred to [2] for some discussion of the effect of m on be-BCQ's.

8 Reduced-Complexity Almost-Optimal Search

Motivation

Consider a block-constrained quantizer BCQ($\underline{q}, \underline{l}, n, L$), and let $\bar{Q}(\cdot)$ be the nearest-neighbor, scalar quantization rule corresponding to $\underline{q}$. Given a source vector $\underline{x}$, if

$l(\bar{Q}(\underline{x})) \leq L$, then $\bar{Q}(\underline{x})$ solves the minimization in (3). Therefore, in practice, nearest-neighbor quantization is performed first and the BCQ search is performed only if the resulting total length exceeds L. With this in mind, we ask the question: Can we make use of $\bar{Q}(\underline{x})$ to simplify the BCQ search? In this section, we answer this question in the affirmative.

Differential search

The basic idea is that given a source sequence $\underline{x} = (x_1, x_2, \ldots, x_n)$, equation (3) has an optimal solution $\hat{\underline{y}}$ whose state is close to that of the output $\underline{y}^0 = \bar{Q}(\underline{x})$ of the nearest neighbor quantizer at each time k. We call $\bar{Q}(\underline{x})$ the *unconstrained output.* Specifically, we show in Appendix C that (3) has an optimum solution $\underline{y}^*$ such that

$$s_k^0 - (l(\underline{y}^0) - L + l_{\max} - l_{\min}) < s_k \leq s_k^0, \;\; k = 1, 2, \ldots, n, \tag{12}$$

where $s_k^0 = s(y_1^0, y_2^0, \ldots, y_k^0) \stackrel{\triangle}{=} \sum_{i=1}^k l(y_i^0)$ and $s_k = s(y_1^*, y_2^*, \ldots, y_k^*) \stackrel{\triangle}{=} \sum_{i=1}^k l(y_i^*)$. It follows, then, that at any time k, we can restrict attention (in the dynamic programming search) to a subset of $(l(\underline{y}^0) - L + l_{\max} - l_{\min})$ states determined by s_k^0 and the *excess length* $\tilde{l}(\underline{x}) \stackrel{\triangle}{=} l(\underline{y}^0) - L$.

Since the storage and number of arithmetic operations required by the dynamic programming is proportional to the number of states that need to be considered, the complexity of the search has now been reduced by the factor $L/(\tilde{l}(\underline{x}) + l_{\max} - l_{\min})$. If, as usually happens, $\tilde{l}(\underline{x})$ is small, this is a sizable reduction. On the other hand, $\tilde{l}(\underline{x})$ will occasionally be quite large, and for most applications the cost of implementation is determined by the maximum required storage and number of arithmetic operations. Therefore, it is necessary to put an upper limit on the number of states that will be considered. Specifically, we fix a number $\tilde{L}$ that is much smaller than L but large

enough that $\tilde{l}(\underline{x})$ is no larger than $\tilde{L}$ most of the time. When $\tilde{l}(\underline{x})$ is less than $\tilde{L}$, we restrict attention to the subset, described above, containing no more than $\tilde{L} + l_{\max} - l_{\min}$ states. When $\tilde{l}(\underline{x}) > \tilde{L}$, we use some simple suboptimal strategy; for example in the case of pe-BCQ, we use the naive approach described in Section 3. A better strategy would be to use one of the simple suboptimal methods described in Sections 9 and 10.

An interesting way to look at the above results from defining the *differential state* as

$$s'_k = s^0_k - s_k.$$

Then, we see that $s'_k \in \{0, 1, \ldots, \tilde{l} + l_{\max} - l_{\min} - 1\}$. Therefore, we call the search performed as above the *differential search* in contrast to the search described in Section 5 which we will, henceforth, call the *direct search.* As before, the differential search can be viewed as a search over a *differential trellis* which has, at depth k, a state corresponding to each possible reduction in length from the length of $(y^0_1, y^0_2, \ldots, y^0_k)$, i.e. state s' at depth k represents all k-tuples $(y_1, y_2, \ldots, y_k)$ of quantization levels which satisfy $\sum_{i=1}^{k} l(y^0_i) - l(y_i) = s'$ and every state has at most m branches entering it, where the branch labeled with q_j (or $(x_k - q_j)^2$) comes from the state $s' - (l(y^0_k) - l_j)$ at depth $k-1$. Figure 6 shows a section of a typical differential trellis. Table 3 summarizes the complexity of the differential search in comparison to the direct search, as well as two suboptimal methods to be introduced later.

It remains to demonstrate that $\tilde{L}$ can be chosen so large that most of the time, $\tilde{l}(\underline{x})$ is no larger, but $\tilde{L}$ is small enough to adequately limit the complexity. Figure 7 shows how the performance of the differential search varies with $(\tilde{L} + m)/L$ for the pe-BCQ's considered above for an IID Gaussian source, rate 3 and several dimensions.

We assume that when $\tilde{l}(\underline{x})$ exceeds $\tilde{L}$, the naive approach of Section 3 is used. In this case, the maximum number of states in the differential trellis is $\tilde{L}+l_{\max}-l_{\min} \leq \tilde{L}+m$. Thus, $(\tilde{L}+m)/L$ measures the fractional reduction in complexity resulting from the differential search. In each case, the performance of the differential search improves with increasing $\tilde{L}$ and "saturates" at the performance of the direct search. We also see that the value of $\tilde{L}/L$ needed to insure negligible loss (relative to the direct search) decreases with n, which indicates that the savings in complexity increase with n. For example, for $n = 192$, choosing $\tilde{L} = 70$, yields about a 6-fold reduction in complexity with .03 dB loss in signal-to-noise ratio.

Some idea of why it is possible to choose $\tilde{L}$ much smaller than L is gained from the following lemma which is proved in Appendix D.

Lemma 1 *Given lengths $\underline{l}$ such that $l_{\min} < l_{\max}$, a quantization rule $Q(\cdot)$, and $0 < \alpha < 1$, then for any IID source $\{X_i\}$, there exists a constant K (usually reasonably small) such that for all sufficiently large n (usually not very large),*

$$\Pr\left\{\sum_{i=1}^{n} l(Q(X_i)) - L \geq K\sqrt{n} + n(\bar{l} - \frac{L}{n})\right\} \leq \alpha,$$

for any L, where $\bar{l} = \mathrm{E}l(Q(\underline{X}))$. □

To study the choice of $\tilde{L}$, we apply the above lemma to the nearest-neighbor quantization rule $\bar{Q}(\cdot)$, upon which the differential search is based. We see that choosing α small and $\tilde{L} = K\sqrt{n} + n(\bar{l} - L/n)$ suffices to guarantee that $\tilde{l}(\underline{X})$ is not larger than $\tilde{L}$ most of the time. Now, we examine how this choice of $\tilde{L}$ compares to L. If $\bar{l} \leq L/n$, then $\tilde{L} \leq K\sqrt{n}$, whereas L is ordinarily chosen to be proportional to n. However, if $\bar{l} > L/n$ (and this is often the case), it generally happens that $\bar{l}$ is close enough to L/n that for moderately large values of n, the term $K\sqrt{n}$ dominates and $\tilde{L}$ is again much less than L, as confirmed by the results in Figure 7.

On the other hand, for very large values of n, we need not be concerned. For one thing, the naive approach described in Section 3 works well and it has trivial complexity. Moreover, as indicated in Figure 4, the naive approach works best with optimized thresholds for which $\bar{l}$ is smaller than for nearest-neighbor quantization. This suggests a compromise where a differential search is based on a quantization rule with $\bar{l}$ close enough to L/n that $\tilde{L}$ can be chosen small relative to L. So, by the above lemma, the differential search will rarely give up (i.e. $\tilde{l}(\underline{X}) > \tilde{L}$ with small probability). Of course, in this case, the differential search is no longer guaranteed to find an optimum solution for (3), since the nearest-neighbor quantization rule is not used. However, this is not expected to degrade the performance significantly.

9 Low-Complexity Greedy Search

The previous section illustrated the value of using the unconstrained output $\underline{y}^0$ to guide the search. Another way to use the unconstrained output is the following: Given $\underline{x}$ and $\underline{y}^0 = \bar{Q}(\underline{x})$ such that $l(\underline{y}^0) > L$, one may perturb $\underline{y}^0$ by changing one or more of its components, e.g. y_i^0, to quantization levels that are assigned shorter lengths, while taking care not to cause a large increase in distortion. Of course, the dynamic programming discussed previously does this optimally. However, in this section, we describe much simpler suboptimal methods that work nearly as well.

Consider a greedy way to perturb the unconstrained output. The basic idea is to find the component of $\underline{y}^0$ which can be replaced by another quantization level of smaller length in such a way that the reduction in length per unit distortion increase is maximized, i.e. we are looking for the most return in terms of length reduction from the increase in distortion that we incur. Equivalently, one may perturb the

component of $\underline{y}^0$ that minimizes the distortion increase per unit length saved. After removing this component from consideration, this process is repeated until the total length has been reduced to L or all the components of $\underline{y}^0$ are exhausted.

We describe the method in more detail. Define

$$\Delta_i^j \triangleq \begin{cases} \frac{(x_i-q_j)^2-(x_i-y_i^0)^2}{l(y_i^0)-l(q_j)}, & \text{if } l(y_i^0) > l(q_j) \\ +\infty, & \text{otherwise.} \end{cases}$$

Note that Δ_i^j is the increase in distortion per unit of length saved if y_i^0 is replaced by q_j. Then, the *greedy search* is described as follows: For $i = 1, 2, \ldots, n$, the algorithm finds the value of j (call it $\hat{\jmath}_i$) that minimizes Δ_i^j. Then, for every iteration, the value of i (call it $\hat{\imath}$) that minimizes $\Delta_i^{\hat{\jmath}_i}$ is found. The $\hat{\imath}$-th component y_i^0 of $\underline{y}^0$ is, then, replaced by $q_{\hat{\jmath}_i}$ and this component is removed from consideration for future iterations. This process continues until the total length of the perturbed $\underline{y}^0$ is no larger than L, the number of iterations reached some specified maximum $\tilde{N}$, or no more components of $\underline{y}^0$ are available for change (i.e. every component has been visited by the algorithm or has a length of $l_{\min}$ that cannot be reduced).

In the above algorithm, there are two cases where the algorithm stops without producing a valid $\underline{y}$ (i.e. $l(\underline{y}) \leq L$). One case occurs when $\tilde{N}$ iterations are not enough to reduce the length to L. The limit $\tilde{N}$ can be carefully chosen to ensure that this occurs with low probability. Another case occurs when the algorithm would fail to reduce the length to L even if it were to make changes at all possible components. This is possible because the algorithm is allowed to make a change only once in any component. Moreover, the tentative change in a component i is determined aperiori in a greedy fashion. It can be argued that this kind of failure occurs with very low probability, and the simulations verified this in all cases tried. In case of failure, some

suitable action has to be taken, e.g. for the case of pe-BCQ, the naive approach may be used.

There are a variety of ways to actually organize the computations involved in the greedy search, with various tradeoff between storage and computation. However, to give a feeling for its complexity, we mention that one straightforward implementation requires approximately $5m$ arithmetic operations per source sample, $2m + \tilde{N}$ binary comparisons per source sample and $3(n + m)$ storage locations. These are much less than for direct or differential search provided that $\tilde{N}$ is not too large, which is normally the case, as seen from Figure 8. Table 3 summarizes the complexity of several search methods, including the greedy search.

The design and optimization approaches described for the direct search can also be applied here, with the main difference being that the level optimization algorithm described previously no longer guarantees a monotonic decrease in distortion. If n is not too large, a clever approach is to optimize the direct search system for a given n and use the optimum parameters with the greedy search provided n is not too large. This gives excellent results. If n is very large, we use the resulting parameters for the direct search with the largest n that we can run. Further, "fine tuning" can be attempted by trying the optimization methods mentioned above.

Figure 8 shows the performance of the greedy search applied to optimum pe-BCQ systems obtained for the direct search case for an IID Gaussian source and rate 3. The figure shows the variation in SNR as the value of $\tilde{N}$ is varied. We see that the performance improves with increasing $\tilde{N}$ and it saturates at a value of $\tilde{N}$ much less than n. For example, for $n = 192$, saturation occurs at $\tilde{N} = 60$. This gives $\tilde{N}/n \approx .31$, and due to the simplicity of the greedy search, this means a tremendous reduction

in complexity compared to the direct search. Moreover, we observe that the value of $\tilde{N}/n$ where saturation occurs decreases with n.

Finally, comparing the "saturation" levels in Figures 7 and 8, we see that the greedy search works very well compared to the differential (or direct) search, e.g. for $n = 192$, the greedy search achieves SNR of 16.02 dB vs. 16.08 dB for the direct search. It should be noted that, contrary to the case of the differential search, the "saturation" of the greedy search does not necessarily correspond to the performance of the direct search. This is due to the suboptimality of the greedy search.

10 A Lagrange-Multiplier-Based (LM-Based) Suboptimal Search

Another low complexity method is based on the following lemma, which shows that BCQ effectively does a scalar quantization but the scalar quantizer that is used depends on the input source vector.

Lemma 2 *Let $\underline{x}$ be a source vector and $\underline{y}$ the corresponding output of $BCQ(\underline{q}, \underline{l}, n, L)$. Then, there exists a set of thresholds $\underline{t}$ such that*

$$\|\underline{x} - \underline{y}\|^2 = \|\underline{x} - Q_{\underline{t}}(\underline{x})\|^2$$

and

$$l(Q_{\underline{t}}(\underline{x})) \leq l(\underline{y}).$$

□

Proof : First we show, by contradiction, that $x_i < x_j$ implies that $y_i \leq y_j$. Suppose, $x_i < x_j$ but $y_i > y_j$. Let $\underline{z} = (z_1, z_2, \ldots, z_n)$ be such that $z_i = y_j$, $z_j = y_i$ and

$z_k = y_k$, $k \notin \{i, j\}$. Then, $\sum_i l(z_i) = \sum_i l(y_i) \leq L$ and

$$\begin{aligned} \|\underline{x} - \underline{z}\|^2 - \|\underline{x} - \underline{y}\|^2 &= (x_i - y_j)^2 + (x_j - y_i)^2 - (x_i - y_i)^2 - (x_j - y_j)^2 \\ &= 2(y_j - y_i)(x_j - x_i) \\ &< 0, \end{aligned}$$

which contradicts the fact that $\underline{y}$ minimizes $\|\underline{x} - \underline{y}\|^2$ subject to $\sum_i l(y_i) \leq L$. The proof is, now, completed by sorting the source samples in ascending order and grouping the ones which are mapped to the same output levels (which occur in clusters by the result just proven) and setting thresholds between the clusters. If there is a common point between two clusters (an event of probability zero for a continuous source), then it can be assigned to the level with shortest length or arbitrarily if the two levels have the same length. ■

The lemma suggests that for given levels $\underline{q}$ and lengths $\underline{l}$, one might try to quantize a source vector $\underline{x}$ by finding the thresholds of the scalar quantizer that gives an equally good output as BCQ$(\underline{q}, \underline{l}, n, L)$ and using them to perform scalar quantization of $\underline{x}$. Finding such a scalar quantizer might be a complex task. However, a good approximation might suffice. Another approach is to have a collection of quantizer threshold sets, to quantize the source samples using each and to choose the output that gives minimum distortion with total length no larger than L. In this approach, a good collection of threshold sets is needed. In the following, we show how to choose such a collection that is indexed by just one parameter.

Motivated by the Lagrange-multiplier formulation of entropy-constrained, scalar quantization and by the work of Chou, *et al.*[11], we consider scalar quantizer thresholds that minimize the quantity $(x - q_j)^2 + \lambda l_j$ for different values of the Lagrange multiplier λ. To further motivate this collection of scalar quantizers, consider the

Lagrangian relaxation [17] of (3):

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n}(x_i - y_i)^2 + \lambda l(y_i), \\ \text{subject to} \quad & y_i \in \{q_1, q_2, \ldots, q_m\} \ \text{ for all } i, \end{aligned} \tag{13}$$

where $\lambda \geq 0$ is a Lagrange multiplier. A well-known fact from non-linear programming theory is that a solution $\underline{y}(\lambda)$ to (13) solves (3) with $L = L(\lambda) \stackrel{\triangle}{=} \sum_i l(y_i(\lambda))$. This suggests that a possible way to solve (3) is to solve (13) for several values of λ and keep the solution that results in the smallest distortion $D(\lambda) \stackrel{\triangle}{=} \sum_i (x_i - y_i(\lambda))^2$ with $L(\lambda) \leq L$.

It can be easily seen that a solution to (13) is given by:

$$y_i(\lambda) \ = \ q_j,$$

where

$$(x_i - q_j)^2 + \lambda l_j \leq (x_i - q_k)^2 + \lambda l_k, \text{ for all } k \in \{1, 2, \ldots, m\},$$

$i = 1, 2, \ldots, n$, where ties are broken arbitrarily, i.e. solving (13) is equivalent to performing scalar quantization on $\underline{x}$ with a quantizer that minimizes the quantity $(x_i - q_j)^2 + \lambda l_j$. It is shown in Appendix E that, for $\lambda \geq 0$, $L(\lambda)$ is non-increasing with λ and that $D(\lambda)$ is non-decreasing with λ. Moreover, it is easy to see that $L(\lambda) = n l_{\min}$ for sufficiently large λ.

Finally, the scalar quantization to minimize $(x_i - q_j)^2 + \lambda l_j$ can be performed using thresholds $\underline{t}(\lambda) = \{t_1, t_2, \ldots, t_{m-1}\}$, given by

$$\begin{aligned} t_1 \ &= \ \min\left\{\frac{q_1 + q_k}{2} + \frac{\lambda}{2}\frac{l_k - l_1}{q_k - q_1} : k > 1\right\}, \\ t_j \ &= \ \max\left\{t_{j-1}, \min\{\frac{q_j + q_k}{2} + \frac{\lambda}{2}\frac{l_k - l_j}{q_k - q_j} : k > j\}\right\}, j = 2, 3, \ldots, m-1. \end{aligned} \tag{14}$$

Therefore, we can try to solve (3) by solving (13) for values of λ starting at zero and increasing gradually until $L(\lambda) \leq L$. If $L(\lambda) = L$, the solution obtained is optimum. Otherwise, it might not be. However, it would be a good approximation. The goodness of the approximation is a function of the fineness of the step by which λ is varied. In order to limit the number of steps needed, we might start with a larger step and then reduce the step size to get a finer approximation. And to limit the complexity, we limit the number values of λ to be tried. The performance of such an algorithm for an IID Gaussian source is shown in Figure 9, where λ is varied in steps of 0.1 and if the total length gets below L, the step size is divided by 10 an so on, until the total length equals L or some maximum allowable number of iterations $\hat{N}$ is exceeded. We did not spend any effort in optimizing the strategy of varying λ. Methods from non-linear programming can also be used to optimize the strategy, e.g. bisection and related methods (see [18]). Also, it should be noted that in the above method, iterations continue until $L(\lambda) = L$ or the maximum number of iterations is exceeded. In many cases, no improvement is obtained after a certain number of iterations. Thus, better stopping criteria are possible and need to be, further, investigated.

Finally, we comment on the complexity of the above method. For each iteration (i.e. for each value of λ we try), the thresholds $\underline{t}(\lambda)$ have to be calculated. Since $\underline{q}$ and $\underline{l}$ are fixed, the quantities $(q_j+q_k)/2$ and $(l_k-l_j)/(2(q_k-q_j))$, $j,k \in \{1,2,\ldots,n\}$, $k>j$ in (14) can be calculated off-line and stored. This requires $m(m-1)$ storage locations. Then, the calculation of $\underline{t}(\lambda)$ needs no more than $m(m-1)$ arithmetic operations and $m(m-1)/2$ binary comparisons. Once the thresholds $\underline{t}(\lambda)$ are found, the source samples are quantized using these thresholds. This, in turn, requires no more than

$n(m-1)$ binary comparisons and computing the total length requires no more than n arithmetic operation. For $\hat{N}$ iterations, the method performs, approximately, $m(m-1)\hat{N}/n + \hat{N}$ arithmetic operations per source sample and $(m-1)(m+2)\hat{N}/2n$ binary comparisons per source sample. Therefore, the complexity is controlled by $\hat{N}$.

Figure 9 shows the performance of this method applied to pe-BCQ with rate 3 for an IID Gaussian source. As in the case of the greedy search, we notice that the performance improves with increasing $\hat{N}$ and saturates at values of $\hat{N}$ much less than n. For example, for $n = 192$, this method achieves a performance within .03 dB of the direct search performance with $\hat{N} = 20$. Again, this is a tremendous reduction in complexity. It is even simpler than the greedy search. Indeed, from Table 3, we see that, in contrast to the greedy search, the LM-based search needs storage independent of n. Moreover, for large n, the arithmetic operations dominate the complexity of the LM-based method while the binary comparisons dominate the complexity of the greedy search. So, the two methods have features that satisfy different needs. Finally, comparing Figures 8 and 9, we see that in general, the LM-based method performs better than the greedy method. The reason is that, most of the time, the LM-based method produces an optimal or near-optimal solution, while the greedy method has no such optimality features.

A final remark is that, in contrast to the direct and differential search methods, the greedy and LM-based methods work for any values of $\underline{l}$, i.e. they do not require $\underline{l}$ to be rational.

11 A Node-Varying, Block-Constrained Quantizer

In this section, we propose a method to reduce the code-space loss of pe-BCQ and, thereby, improve its performance. The basic idea is introduced by means of an example.

Consider pe-BCQ with dimension $n = 6$, rate $r = 2$, threshold $L = nr = 12$, quantization levels $\underline{q} = (q_1, q_2, \ldots, q_5)$ (whose specific values are of little concern in this discussion), and prefix code $C = \{0, 10, 110, 1110, 1111\}$ with lengths $\underline{l} = (1, 2, 3, 4, 4)$. Figure 10 shows the corresponding trellis[8]. A bold branch (solid or dashed) indicates that this state transition corresponds to two codewords of the same length (in this case 1110 and 1111). A dotted branch is a *useless branch*; i.e. one that is not used in any BCQ codevector. For example, the node corresponding to state 6 at depth 2 (called "node $(6, 2)$" from now on) has one bold dotted branch emerging from it, due to the fact that any path from this node that begins with a codeword of length 4 will have length at least $6 + 4 + 1 + 1 + 1 + 1 = 13$. Since this is greater than the threshold, no such path corresponds to a codevector in the BCQ codebook. On the other hand, it is possible to have paths beginning with codewords of length 1, 2 or 3 leaving this node. Thus, three solid branches are shown.

This BCQ has 1107 codevectors and, consequently, code-space loss $(L - \log_2 \left|\mathcal{C}_{\text{bcq}}(\underline{q}, \underline{l}, n, L)\right|)/n = (12 - \log_2 1107)/6 = .31$, which is fairly large. In fact, the useless branches are a substantial cause of this loss. Specifically, a useless branch wastes bits because if it were not assigned a codeword, then shorter codewords could be assigned to the other branches stemming from the same node. The new BCQ

[8]See the discussion in Section 5 for a description of the trellis.

codebook would have more codevectors employing these other branches. Hence, the codebook would be enlarged, the distortion would be decreased, but the rate $r = L/n$ would remain the same.

In this section, we explore such *node-varying* BCQ's (nv-BCQ), in which each node of the trellis is assigned a subset of the original quantization levels (or the original set itself) and a prefix code for just these levels. It should be noted that just as with ordinary pe-BCQ, dynamic programming can be used to perform the trellis search, because the level subset and prefix code to be used at any given node is uniquely determined by the node. Moreover, the binary sequence produced by any path on such a trellis is uniquely decodable, because the prefix code to be used at given node is uniquely determined by the node, and the node is uniquely determined from previously decoded symbols.

As a concrete example, suppose node $(6, 2)$ is assigned the level subset $\{q_1, q_2, q_3\}$ and the prefix code $\{0, 10, 11\}$. Further, suppose all other nodes are assigned the full set of levels and the original prefix code C. One may easily check that the resulting nv-BCQ codebook contains the original pe-BCQ codebook plus additional codevectors, such as the level sequence $(q_4, q_2, q_3, q_2, q_1, q_1)$ whose length has been reduced from 13 to 12. In a sense, the pruning of the branches corresponding to levels q_4 and q_5 has resulted in a "denser" trellis, in somewhat the same way that careful pruning of a (biological) plant produces a denser plant.

As another example, notice that node $(11, 5)$ has only one useful branch stemming from it, corresponding to level q_1 and the binary codeword 0. Accordingly, this node can be assigned the level subset $\{q_1\}$ and the degenerate prefix code containing only the empty string with zero length. This means that when this node is reached, no

further bits need be sent, because q_1 is necessarily the next level. Unfortunately, while seeming beneficial, this change adds no new codevectors. On the other hand, if we assign the level subset $\{q_1, q_2\}$ and the prefix code {0,1}, then new codevectors will be added to the codebook.

As a final example, consider assigning the subset $\{q_1\}$ and the prefix code containing just the empty string to node $(12, 5)$. In this case, all level sequences that lead to this node and then are followed by q_1 are added to the codebook. One may, similarly, add more codevectors by assigning the subset $\{q_1\}$ and the the empty string codebook to nodes $(12, 3)$ and $(12, 4)$.

A basic principle is that given a pe-BCQ (or, for that matter, an nv-BCQ) and a node in its trellis, if one assigns to this node a subset of the levels that contains all levels presently useful for this node, and one assigns a new prefix code for this subset such that the new codeword assigned to a useful level is no longer than the previous codeword assigned to it, then the new nv-BCQ codebook will contain the previous codebook, the resulting average distortion will be less than or equal to the the present distortion (usually strictly less), and the rate will remain the same.

This basic principal notwithstanding, optimizing an nv-BCQ is a difficult task. One might try nodewise optimization. That is, for any given node, one might try all possible level subsets and all possible prefix codes to find the one that results in the largest codebook or the smallest distortion. But such nodewise optimization is not likely to be the best solution. For example, assigning (as above) the level subset $\{q_1, q_2\}$ and the prefix code $\{0, 1\}$ to node $(11, 5)$ gives the best codebook among all those produceable by changes only at this node. However, it causes the branch from node $(6, 2)$ corresponding to q_4 and q_5 to no longer be useless, which eliminates

the justification for assigning level subset $\{q_1, q_2, q_3\}$ to this node. So it is not clear whether it is better to make the change to node $(11, 5)$, to make the change to node $(6, 2)$ (described previously), or to make both changes.

Moreover, even if an nv-BCQ could be truly optimized, considerable storage would be needed for the level subsets and prefix codes assigned to each of the approximately nL nodes. To reduce storage, one might limit the level subsets and prefix codes to members of some relatively small collections of such. Nevertheless, for each node one would still need to specify the index of some subset and prefix code. To avoid such added storage, in the following we propose the use of a simple rule for assigning level subsets and prefix codes to nodes.

Given scalar quantization levels $\underline{q} = (q_1, q_2, \ldots, q_m)$ with probabilities $p_1 \geq p_2 \geq \ldots \geq p_m$, for $1 \leq k \leq m$, let $\underline{q}^k = (q_1, q_2, \ldots, q_k)$, and let C^k be a Huffman code for probabilities $(p_1, p_2, \ldots, p_k)$. (C^1 contains only the empty string.) Let l_j^k denote the length of the j-th codeword of C^k, and let $l_{\min}^k$ and $l_{\max}^k$ denote the minimum and maximum lengths, respectively. To node (s, i) assign the level subset $\underline{q}^k$ and the prefix code C^k for the largest k such that

$$l_{\max}^k \;\leq\; L - (n-i)l_{\min}^k - s \;.$$

One may easily check that this guarantees that the branches corresponding to $q_1, \ldots, q_k$ will all be useful, assuming that each can be followed by a path whose branches all have length $l_{\min}^k$ or less[9].

Applying this approach to the pe-BCQ example in Figure 10 leads to the nv-BCQ whose trellis is shown in Figure 11. The (nontrivial) prefix codes are $C^2 = \{0, 1\}$, $C^3 = \{0, 10, 11\}$, $C^4 = \{0, 10, 110, 111\}$ and $C^5 = \{0, 10, 110, 1110, 1111\}$. The codebook of

[9] We don't actually look ahead to see if there is such a path because this would add to the complexity.

this nv-BCQ has 2342 codevectors. (approximately twice the original number), and the code-space loss has been reduced to .13.

Table 4 shows the SNR's resulting from applying this nv-BCQ approach to IID Gaussian and Laplacian sources. We chose $m = 20$. To design the quantization levels $q_1, \ldots, q_{20}$, we started with uniformly spaced levels and thresholds, found level probabilities $p_1, \ldots, p_20$, and Huffman codebooks $C^2, C^3, \ldots, C^{20}$ for the level subsets $\underline{q}^2, \underline{q}^3, \ldots, \underline{q}^{20}$. We then found a new set of quantization levels (i.e. centroids) in the manner described in Section 6, and a new set of Huffman codes. This process was iterated until the performance no longer improved significantly.

As shown in Table 4, the performance of such nv-BCQ, is better than that of pe-BCQ. For example, comparing with Table 2, we see that the SNR of nv-BCQ with $n = 128$ is comparable to that of ordinary pe-BCQ with $n = 192$. Moreover, for $n = 48$, the SNR of nv-BCQ is .06-.6 dB higher SNR for the IID Gaussian source than for pe-BCQ with the same n, and .2-.5 dB higher for the IID Laplacian source. The gains tends to be larger at smaller rates, because varying the prefix codes allows the effective average length to be closer to, or even less than, one. Finally, we note that the gains in nv-BCQ come at the expense of only a slight increase in complexity.

The implementation of the suboptimal search methods discussed in Sections 8-10 for the node-varying case is not straightforward and needs to be further investigated. We leave this for future research.

Motivated by the above nv-BCQ, one may define more general classes of node-varying/time-varying BCQ's. As shown in the examples above, there are numerous possible strategies for designing such BCQ's. Such strategies need to be investigated further. Our discussion here does no more than scratch the surface. The richness of

such schemes necessitates deeper understanding. this to future studies. We leave this to future studies.

12 Conclusion

In this paper, we have explored a quite general class of fixed-rate quantizers that we called block-constrained quantizers, motivated by the work of Laroia and Farvardin [2]. We proposed methods to reduce the binary encoding complexities of such quantizers.

We have discussed the relationship between block-constrained quantizers, entropy-coded/entropy-constrained quantizers and permutation codes. Moreover, we have explored some basic properties of BCQ's that give insight into their behavior.

Different methods to reduce the search complexity of BCQ's have also been investigated. In particular, very low-complexity, suboptimal search methods have been proposed and shown to perform very close to the optimal while providing a large reduction in complexity.

Node-varying BCQ's have been proposed as a means of reducing the code-space loss of pe-BCQ's. The results show significant gains.

The investigation of BCQ's is by no means complete. The optimization of BCQ's, particularly the lengths, is still an open problem. The effects of the number of levels need further investigation. Node-varying BCQ's need to be further studied and the application of the low-complexity search methods to node-varying BCQ's is of particular interest.

A Convergence of the Performance of the Naive System

Consider an IID source $\{X_i\}$ with mean μ and finite variance σ^2, and a naive system based on $\underline{q}, \underline{t}, \underline{l}$ with blocklength n and rate r. Let $\underline{X} = (X_1, X_2, \ldots, X_n)$ and $q_{\max} = \max_j |q_j|$. Assume that the decoder produces $Q_{\underline{t}}(\underline{X}) = (Q_{\underline{t}}(X_1), Q_{\underline{t}}(X_2), \ldots, Q_{\underline{t}}(X_n))$ if $l(Q_{\underline{t}}(\underline{X})) \leq nr$ and, otherwise, produces $(\mu, \mu, \ldots, \mu)$. We denote the distortion of this naive system by $D_{\text{naive}}(\underline{q}, \underline{t}, \underline{l}, n, nr)$. We will show that

$$\lim_{n\to\infty} D_{\text{naive}}(\underline{q}, \underline{t}, \underline{l}, n, nr) = D_{\text{sq}}(\underline{q}, \underline{t}),$$

if $\bar{l}(\underline{t}, \underline{l}) < r$.

To this end, define

$$A_n \stackrel{\triangle}{=} \{\underline{x} = (x_1, x_2, \ldots, x_n) : \frac{1}{n}\sum_{i=1}^{n} l(Q_{\underline{t}}(x_i)) \leq r\}.$$

and let $\underline{t}$, $\underline{l}$ and r be such that $\bar{l}(\underline{t}, \underline{l}) < r$. Then, by the weak law of large numbers, $\mathrm{P}(A_n^{\mathrm{c}}) \stackrel{\triangle}{=} \Pr\{\underline{X} \in A_n^{\mathrm{c}}\} \to 0$ as $n \to \infty$.

Now,

$$D_{\text{naive}}(\underline{q}, \underline{t}, \underline{l}, n, nr) = \frac{1}{n}\int_{A_n} \|\underline{x} - Q_{\underline{t}}(\underline{x})\|^2 p(\underline{x}) d\underline{x} + \frac{1}{n}\int_{A_n^{\mathrm{c}}} \|\underline{x} - \underline{\mu}\|^2 p(\underline{x}) d\underline{x},$$

where $\underline{\mu} = (\mu, \mu, \ldots, \mu)$. It follows that

$$\begin{aligned} D_{\text{naive}}(\underline{q}, \underline{t}, \underline{l}, n, nr) &= D_{\text{sq}}(\underline{q}, \underline{t}) + \frac{1}{n}\int_{A_n^{\mathrm{c}}} \left(\|\underline{x} - \underline{\mu}\|^2 - \|\underline{x} - Q_{\underline{t}}(\underline{x})\|^2\right) p(\underline{x})\, d\underline{x} \\ &= D_{\text{sq}}(\underline{q}, \underline{t}) + \mathrm{E}[f_n(\underline{X})], \end{aligned}$$

where

$$f_n(\underline{x}) \stackrel{\triangle}{=} \frac{\|\underline{x} - \underline{\mu}\|^2 - \|\underline{x} - Q_{\underline{t}}(\underline{x})\|^2}{n} I_{A_n^{\mathrm{c}}}(\underline{x}),$$

and I_A is the indicator function. It suffices to show that $E[f_n(\underline{X})] \to 0$ as $n \to \infty$. We can easily see that

$$f_n(\underline{X}) \to 0, \text{ in probability, as } n \to \infty,$$

$$|f_n(\underline{X})| \leq g_n(\underline{X}) \stackrel{\triangle}{=} \frac{2\|\underline{X}-\underline{\mu}\|^2+n(\mu^2+q_{\max}^2)}{n},$$

$$\mathrm{E}[g_n(\underline{X})] = 2\sigma^2 + \mu^2 + q_{\max}^2 < \infty, \text{ for all } n,$$

and

$$g_n(\underline{X}) \to 2\sigma^2 + \mu^2 + q_{\max}^2, \text{ with probability 1, as } n \to \infty.$$

Define $a_n \stackrel{\triangle}{=} \mathrm{E}[f_n(\underline{X})]$. First, we show that for any convergent subsequence a_{n_k}, $a_{n_k} \to 0$ as $k \to \infty$. Note that $f_{n_k}(\underline{X}) \to 0$, in probability, as $k \to \infty$. Then, by Proposition 4.18 of [19], there exists a further subsequence $f_{n_{k_i}}(\underline{X})$ such that $f_{n_{k_i}}(\underline{X}) \to 0$, with probability 1, as $i \to \infty$. Now, by Theorem 4.17 of [19], $a_{n_{k_i}} = \mathrm{E}[f_{n_{k_i}}(\underline{X})] \to 0$ as $i \to \infty$ and since a_{n_k} is convergent, then $a_{n_k} \to 0$ as $k \to \infty$.

Now, it suffices to show that $a_n \to 0$ as $n \to \infty$. To show this, note that a_n is bounded. Thus, for any subsequence a_{n_k}, there exists a convergent subsequence $a_{n_{k_i}}$ and by the result above, $a_{n_{k_i}} \to 0$ as $i \to \infty$. Therefore, any subsequence of a_n has a further subsequence that converges to 0. A standard result in real analysis implies that $a_n \to 0$ as $n \to \infty$. ■

B Proof of (8)

Consider $\underline{l}$ satisfying Kraft's inequality and $L < nl_{\max}$. Then

$$\begin{aligned} 1 &\geq (\sum_{j=1}^{m} 2^{-l_j})^n \\ &= \sum_a n_a 2^{-a}, \end{aligned}$$

where n_a is the number of sequences of n quantization levels having total length a. Now,

$$\begin{aligned}\sum_a n_a 2^{-a} &= \sum_{a \le L} n_a 2^{-a} + \sum_{a > L} n_a 2^{-a}, \\ &> \sum_{a \le L} n_a 2^{-a} \\ &\ge 2^{-L} \sum_{a \le L} n_a \\ &= 2^{-L} \left| \mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L) \right|,\end{aligned}$$

where the first inequality follows since $nl_{\max} > L$ and the last equality follows from the definitions of n_a and $\mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L)$. Therefore,

$$\left| \mathcal{C}_{\rm bcq}(\underline{q}, \underline{l}, n, L) \right| < 2^L.$$

The result (8) follows by taking $\log_2$ of both sides and dividing by n. ■

C Proof of (12)

Let $\underline{y}^0 = \bar{Q}(\underline{x})$ and $\tilde{l} = l(\underline{y}^0) - L$. If $l(\underline{y}^0) \le L$, then $\underline{y}^* = \underline{y}^0$ satisfies (12). Thus, we assume $l(\underline{y}^0) > L$. First, we observe that there always exists an optimum solution $\underline{\hat{y}}$ to (3) such that $l(\hat{y}_i) \le l(y_i^0)$ for all i. This can be seen as follows: Suppose $\underline{\check{y}}$ is an optimum solution to (3). Let $\underline{\hat{y}}$ be such that $\hat{y}_i = \check{y}_i$ if $l(\check{y}_i) \le l(y_i^0)$ and $\hat{y}_i = y_i^0$, otherwise. Then, we can easily see that $\underline{\hat{y}}$ satisfies the hypothesis, i.e.

$$\sum_{i=1}^{n} l(\hat{y}_i) \le L,$$

and

$$\|\underline{x} - \underline{\hat{y}}\|^2 \le \|\underline{x} - \underline{\check{y}}\|^2.$$

Now, let $\underline{\hat{y}}$ be as above, then $\sum_{i=1}^{k}(l(y_i^0) - l(\hat{y}_i))$ is non-decreasing in k and $\sum_{i=1}^{n}(l(y_i^0) - l(\hat{y}_i)) \geq \tilde{l}$. Therefore, there exists K, $1 \leq K \leq n$, such that

$$\sum_{i=1}^{K-1} l(y_i^0) - l(\hat{y}_i) < \tilde{l} \leq \sum_{i=1}^{K} l(y_i^0) - l(\hat{y}_i).$$

Moreover, $\underline{y}^* = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_K, y_{K+1}^0, \ldots, y_n^0)$ also solves (3) and for any $1 \leq k \leq n$,

$$\sum_{i=1}^{k} l(y_i^0) - l(y_i^*) \quad < \quad \tilde{l} + l_{\max} - l_{\min}.$$

Also, since $l(\hat{y}_i) \leq l(y_i^0)$, we have

$$\sum_{i=1}^{k} l(y_i^*) \quad \leq \quad \sum_{i=1}^{k} l(y_i^0).$$

The above two equations imply that $\underline{y}^*$ satisfies (12). ■

D Proof of Lemma 1

Consider a quantization rule $Q(\cdot)$ applied to an IID source $\{X_i\}$ and let the lengths $\underline{l} = (l_1, l_2, \ldots, l_m)$ correspond to the quantization levels $\underline{q} = (q_1, q_2, \ldots, q_m)$, respectively. Without loss of generality, we assume that $\min_j l_j \stackrel{\triangle}{=} l_{\min} < l_{\max} \stackrel{\triangle}{=} \max_j l_j$ and $\Pr\{l(Q(X_1)) = l_j\} \neq 0$ if $l_j \in \{l_{\min}, l_{\max}\}$. Also, let $p_{\max} = \Pr\{l(Q(X_1)) = l_{\max}\}$ and $\bar{l} = \mathrm{E}[l(Q(X_1))]$. Define the moment generating function $M(s)$ by

$$M(s) \stackrel{\triangle}{=} \mathrm{E}\left[\exp\{sl(Q(X_1))\}\right] = \sum_{j=1}^{m} p_j \exp(sl_j),$$

where $p_j \stackrel{\triangle}{=} \Pr\{l(Q(X_1)) = l_j\}$, and define the large deviation rate function $I(u)$ by [20]

$$I(u) \stackrel{\triangle}{=} \begin{cases} \sup_{s \geq 0}(su - \ln M(s)), & u \geq \bar{l} \\ \sup_{s < 0}(su - \ln M(s)), & u < \bar{l} \end{cases}.$$

It can be easily shown that $I(u)$ is convex $\bigcup$ and continuous on $[l_{\min}, l_{\max}]$. Moreover, $I(\bar{l}) = 0$, $I'(\bar{l}) = 0$, and $I(l_{\max}) = \ln(1/p_{\max}) > 0$. Finally, $1/I''(u)$ is bounded. Let $K_1 = \sup_u(1/I''(u)) < \infty$.

Let $\beta = I(l_{\max})$. Now, given $0 < \alpha < 1$ and N large enough that $\ln(1/\alpha)/N \leq \beta$, then by the intermediate value theorem, for every $n \geq N$, there exists $u_n > \bar{l}$ such that $I(u_n) = \ln(1/\alpha)/n$ or $\exp(-nI(u_n)) = \alpha$. By Chernoff's bound,

$$\begin{aligned} \Pr\left\{\sum_{i=1}^{n} l(Q(X_i)) \geq nu_n\right\} &\leq \exp(-nsu_n)M(s)^n, \quad \text{for all } s \geq 0, \\ &= \exp(-n(su_n - \ln M(s))), \quad \text{for all } s \geq 0, \end{aligned}$$

which implies

$$\begin{aligned} \Pr\left\{\sum_{i=1}^{n} l(Q(X_i)) \geq nu_n\right\} &\leq \inf_{s\geq 0} \exp(-n(su_n - \ln M(s))), \\ &= \exp(-nI(u_n)) \\ &= \alpha, \end{aligned}$$

Since $I(\bar{l}) = I'(\bar{l}) = 0$, then by the second-order Taylor's theorem (e.g. [17, p. 504]), there exists γ_n, $\bar{l} < \gamma_n < u_n$, such that

$$I(u_n) = \frac{1}{2}I''(\gamma_n)(u_n - \bar{l})^2,$$

or

$$u_n = \sqrt{\frac{2I(u_n)}{I''(\gamma_n)}} + \bar{l} = \sqrt{\frac{2\ln(1/\alpha)}{I''(\gamma_n)}}\frac{1}{\sqrt{n}} + \bar{l} \leq \sqrt{2\ln(1/\alpha)K_1}\frac{1}{\sqrt{n}} + \bar{l} = \frac{K}{\sqrt{n}} + \bar{l},$$

where $K \stackrel{\triangle}{=} \sqrt{2\ln(1/\alpha)K_1}$. Thus, for any L, we have

$$\begin{aligned} \Pr\left\{\sum_{i=1}^{n} l(Q(X_i)) - L \geq K\sqrt{n} + n(\bar{l} - \frac{L}{n})\right\} &= \Pr\left\{\sum_{i=1}^{n} l(Q(X_i)) \geq n(\frac{K}{\sqrt{n}} + \bar{l})\right\} \\ &\leq \Pr\left\{\sum_{i=1}^{n} l(Q(X_i)) \geq nu_n\right\} \\ &\leq \alpha. \end{aligned}$$

■

E Proof of Properties of $L(\lambda)$ and $D(\lambda)$

It is sufficient to show th following:

Lemma 3 *Let $\tilde{\lambda} > \hat{\lambda} \geq 0$. Given x, let $\tilde{q}$ and $\hat{q}$ be such that*

$$(x-\tilde{q})^2 + \tilde{\lambda} l(\tilde{q}) \leq (x-q)^2 + \tilde{\lambda} l(q), \text{ for all } q \in \underline{q}, \tag{15}$$

$$(x-\hat{q})^2 + \hat{\lambda} l(\hat{q}) \leq (x-q)^2 + \hat{\lambda} l(q), \text{ for all } q \in \underline{q}. \tag{16}$$

Then,

1. $(x-\tilde{q})^2 \geq (x-\hat{q})^2$.

2. $l(\tilde{q}) \leq l(\hat{q})$.

□

Proof :

1. Suppose

$$(x-\tilde{q})^2 < (x-\hat{q})^2. \tag{17}$$

Then,

$$\begin{aligned}
(x-\tilde{q})^2 &\leq (x-\tilde{q})^2 + \hat{\lambda} l(\tilde{q}), \text{ since } \hat{\lambda} \geq 0, l(\tilde{q}) > 0, \\
&< (x-\hat{q})^2 + \hat{\lambda} l(\tilde{q}), \text{ by (17)}, \\
&< (x-\hat{q})^2 + \tilde{\lambda} l(\tilde{q}), \text{ since } \tilde{\lambda} > \hat{\lambda}, l(\tilde{q}) > 0, \\
&\leq (x-\tilde{q})^2 + \hat{\lambda} l(\tilde{q}) - \hat{\lambda} l(\hat{q}) + \tilde{\lambda} l(\tilde{q}), \text{ by (16)}, \\
&= (x-\tilde{q})^2 - (\tilde{\lambda} - \hat{\lambda}) l(\tilde{q}) - \hat{\lambda} l(\hat{q}), \\
&\leq (x-\tilde{q})^2 - \hat{\lambda} l(\hat{q}), \text{ since } \tilde{\lambda} > \hat{\lambda}, l(\tilde{q}) > 0, \\
&\leq (x-\tilde{q})^2, \text{ since } \hat{\lambda} \geq 0, l(\tilde{q}) > 0,
\end{aligned}$$

which is a contradiction.

2. From (15),

$$\begin{aligned} \tilde{\lambda}(l(\tilde{q}) - l(\hat{q})) &\leq (x - \tilde{q})^2 - (x - \hat{q})^2, \\ &\leq 0, \text{ by part 1.} \end{aligned}$$

Since $\tilde{\lambda} > 0$, then $l(\tilde{q}) - l(\hat{q})) \leq 0$, and the result follows.

■

References

[1] N. Farvardin and J. W. Modestino, "Adaptive buffer-instrumented entropy-coded quantizer performance for memoryless sources," *IEEE Trans. Infrom. Theory,* vol. IT-32, pp. 9-22, Jan. 1986.

[2] R. Laroia and N. Farvardin, "A structured fixed-rate vector quantizer derived from a variable-length encoded scalar quantizer," *submitted to IEEE Trans. Infrom. Theory.*

[3] R. Laroia and N. Farvardin, "Extension of of the fixed-rate structured vector quantizer to vector sources," *presented in the twenty-fifth annual conference on information sciences and systems,* Johns Hopkins, Baltimore, Mar. 1991.

[4] T. J. Goblick, Jr. and J. L. Holsinger, "Analog source digitization: A comparison of theory and practice," *IEEE Trans. Inform. Theory,* vol. IT-13, pp. 323-326, Apr. 1967.

[5] H. Gish and J. N. Pierce, "Asymptotically efficient quantizing," *IEEE Trans. Inform. Theory,* vol. IT-14, pp. 676-683, Sept. 1968.

[6] R. C. Wood, "On optimum quantization," *IEEE Trans. Inform. Theory,* vol. IT-15, pp. 248-252, Mar. 1969.

[7] T. Berger, "Optimum quantizers and permutation codes," *IEEE Trans. Inform. Theory,* vol. IT-18, pp. 759-765, Nov. 1972.

[8] A. N. Netravali and R. Saigal, "Optimum quantizer design using a fixed-point algorithm," *Bell Syst. Tech. J.,* vol. 55, pp. 1423-1435, Nov. 1976.

[9] T. Berger, "Minimum entropy quantizers and permutation codes," *IEEE Trans. Inform. Theory,* vol. IT-28, pp. 149-157, Mar. 1982.

[10] N. Farvardin and J. W. Modestino, "Optimum quantizer performance for a class of non-Gaussian memoryless sources," *IEEE Trans. Inform. Theory,* vol. IT-30, pp. 485-497, May 1984.

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

[12] J. C. Kieffer, T. M. Jahns and V. A. Obuljen, "New results on optimal entropy-constrained quantization," *IEEE Trans. Inform. Theory,* vol. IT-34, pp. 1250-1258, Sep. 1988.

[13] T. Berger, F. Jelinek and J. K. Wolf, "Permutation codes for sources," *IEEE Trans. Inform. Theory,* vol. IT-18, pp. 160-169, Jan. 1972.

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

[15] L. C. Stewart, R. M. Gray and Y. Linde, "The design of trellis waveform coders," *IEEE Trans. Comm.,* vol. COM-30, pp. 702-710, Apr. 1982.

[16] A. S. Balamesh and D. L. Neuhoff, *in preparation.*

[17] M. S. Bazaraa and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms,* J. Wiley, 1979.

[18] W. H. Press, et al., *Numerical Recipes: The Art of Scientific Computing.*

[19] H. L. Royden, *Real Analysis,* 3rd Ed., NY:Macmillan, 1988.

[20] J. A. Bucklew, *Large Deviation Techniques in Decision, Simulation, and Estimation,* J. Wiley, 1990.

Table 1: Rough complexity comparisons for lf-BCQ's and pe-BCQ's.

		lf-BCQ	pe-BCQ
Trellis Search	Trellis Depth	n	n
	Number of States	$1.5bnr$	nr
	Branching Factor	m	m
	Operations per Branch	2	2
	Arithmetic Operations per Source Sample	$3bmnr$	$2mnr$
	Storage Locations	$1.5bn^2r$	n^2r
Binary Encoding	Arithmetic Operations per Source Sample	mnr	insignificant
	Storage Locations	bn^3r^2	insignificant

Table 2: SNR, in dB, for pe-BCQ and several other methods.

Rate	pe-BCQ					Entropy-	lf-BCQ
r	$n = 48$	96	144	192	ECSQ	Const. SQ	$n = 32$
IID Gaussian Source							
1.5	6.34 (0.19)	6.48 (0.16)	6.52 (0.15)	6.54 (0.14)	7.16	7.55	7.58
2.0	9.82 (0.063)	10.04 (0.042)	10.12 (0.028)	10.16 (0.021)	10.25	10.55	10.43
2.5	12.91 (0.063)	13.05 (0.031)	13.15 (0.028)	13.22 (0.021)	13.44	13.54	13.21
3.0	15.49 (0.063)	15.86 (0.042)	15.98 (0.035)	16.08 (0.026)	16.35	16.56	16.01
IID Laplacian Source							
1.5	7.18 (0.19)	7.42 (0.16)	7.54 (0.15)	7.59 (0.14)	7.69	8.55	8.22
2.0	10.00 (0.063)	10.51 (0.042)	10.65 (0.028)	10.80 (0.021)	11.26	11.31	10.73
2.5	12.78 (0.15)	13.16 (0.11)	13.33 (0.097)	13.42 (0.094)	14.03	14.32	13.31
3.0	15.79 (0.063)	16.21 (0.042)	16.43 (0.028)	16.52 (0.021)	17.08	17.20	16.05

Table 3: Rough complexity comparisons for different BCQ search methods.

	Direct	Differential	Greedy	LM-based
Trellis Depth	n	n	N.A.	N.A.
Number of States	$L \propto nr$	$\tilde{L} + l_{\max} - l_{\min}$	N.A.	N.A.
Branching Factor	m	m	N.A.	N.A.
Operations per Branch	2	2	N.A.	N.A.
Arithmetic Operations per Source Sample	$2mL$	$2m(\tilde{L} + l_{\max} - l_{\min})$	$5m$	$\frac{m(m-1)\hat{N}}{n} + \hat{N}$
Binary Comparisons per Source Sample			$2m + \tilde{N}$	$\frac{(m-1)(m+2)\hat{N}}{2n}$
Storage Locations	nL	$n(\tilde{L} + l_{\max} - l_{\min})$	$3(n+m)$	$m(m-1)$

Table 4: SNR, in dB, for node-varying BCQ.

Rate	nv-BCQ				pe-BCQ	lf-BCQ	Entropy-
r	$n = 32$	48	64	128	$n = 192$	$n = 32$	Const. SQ
IID Gaussian Source							
1.0	4.40	4.40	4.40	4.40		4.67	4.64
1.5	6.99	6.95	6.92	6.83	6.54	7.58	7.55
2.0	9.92	10.01	10.07	10.16	10.16	10.43	10.55
2.5	12.85	12.97	13.07	13.20	13.22	13.21	13.54
3.0	15.52	15.68	15.82	15.99	16.08	16.00	16.56
IID Laplacian Source							
1.0	3.05	3.06	3.06	3.06		5.61	5.76
1.5	7.55	7.64	7.66	7.70	7.59	8.22	8.55
2.0	10.56	10.77	10.86	11.03	10.80	10.73	11.31
2.5	12.67	12.94	13.04	13.30	13.42	13.31	14.32
3.0	15.64	15.97	16.15	16.44	16.52	16.05	17.20

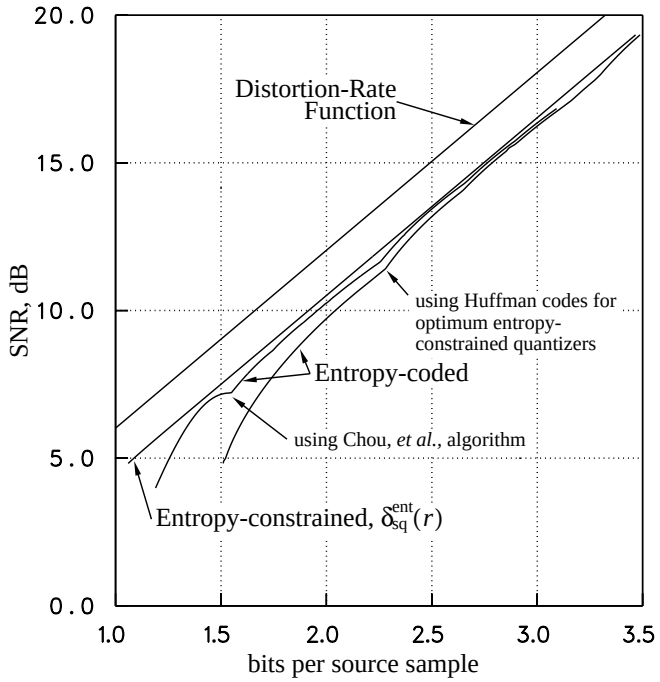

Figure 1: Performance of entropy-coded and entropy-constrained scalar quantizers for an IID Gaussian source.

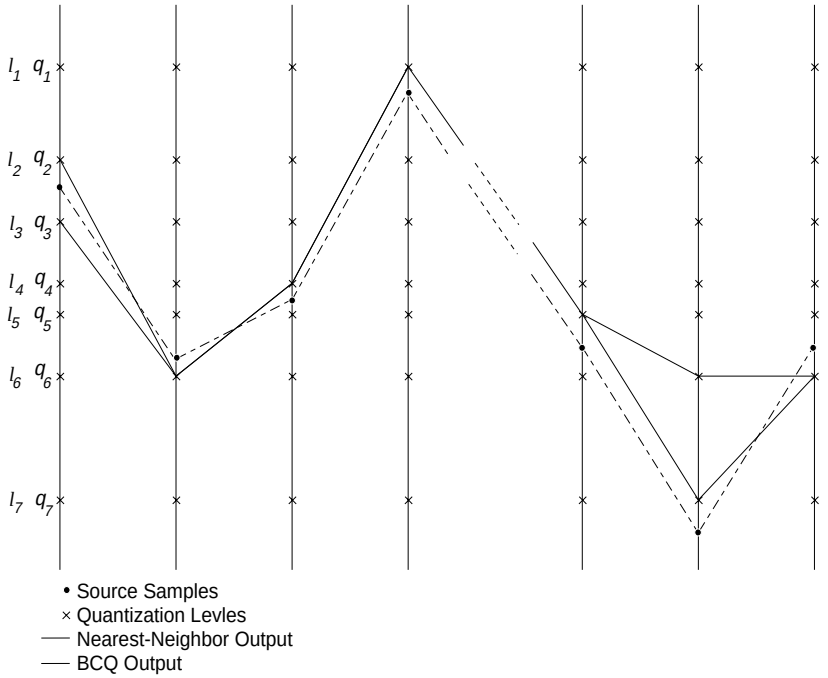

Figure 2: Relationship between nearest-neighbor quantization and BCQ.

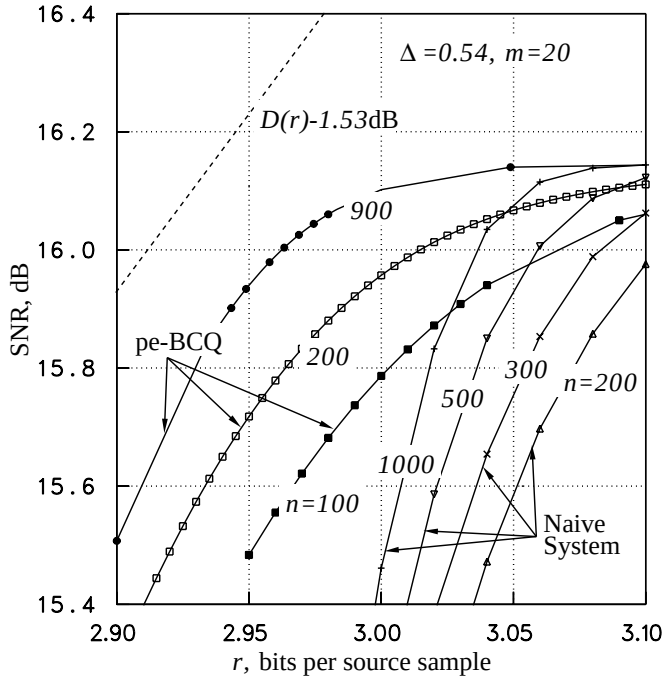

Figure 3: Performance of BCQ and the naive system for fixed $\underline{q}$ and $\underline{l}$ and several values of n.

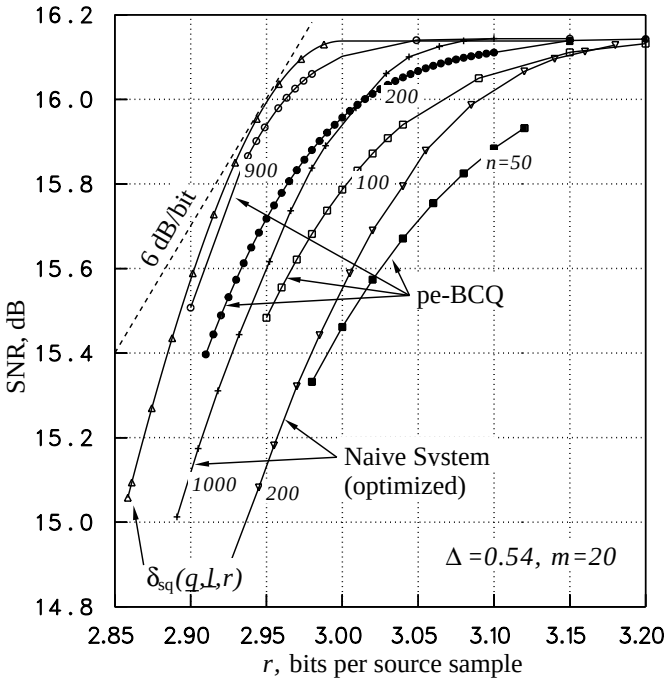

Figure 4: Performance of pe-BCQ and optimized naive system for an IID Gaussian source.

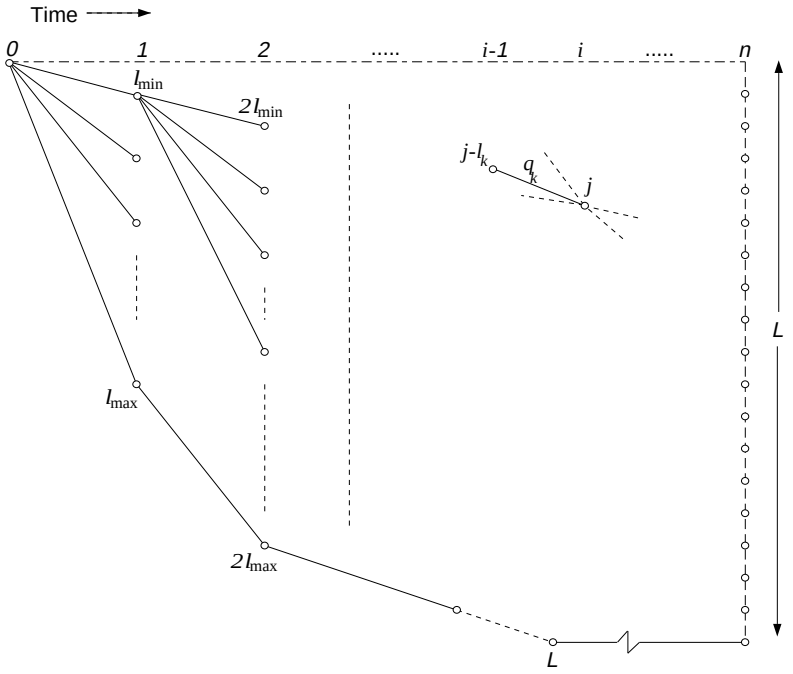

Figure 5: Trellis for the dynamic programming search.

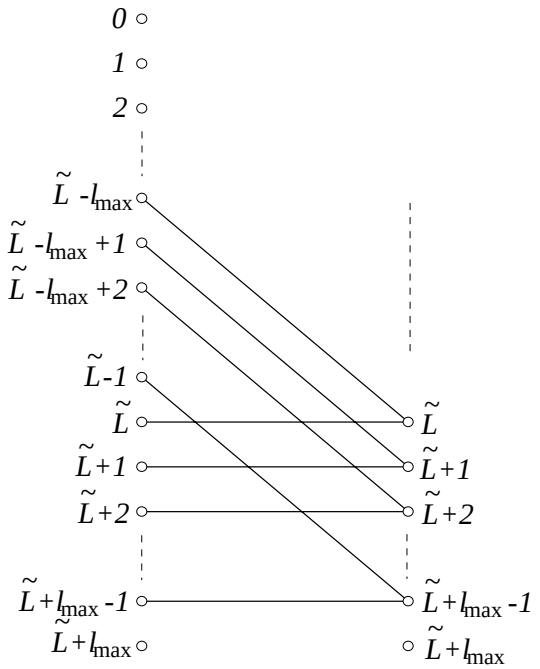

Figure 6: A section of a typical differential trellis.

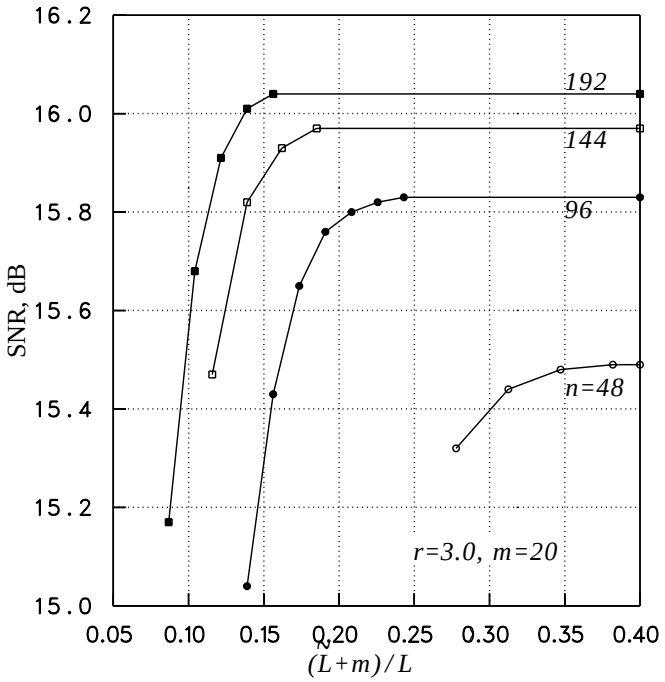

Figure 7: Performance of differential search for an IID Gaussian source.

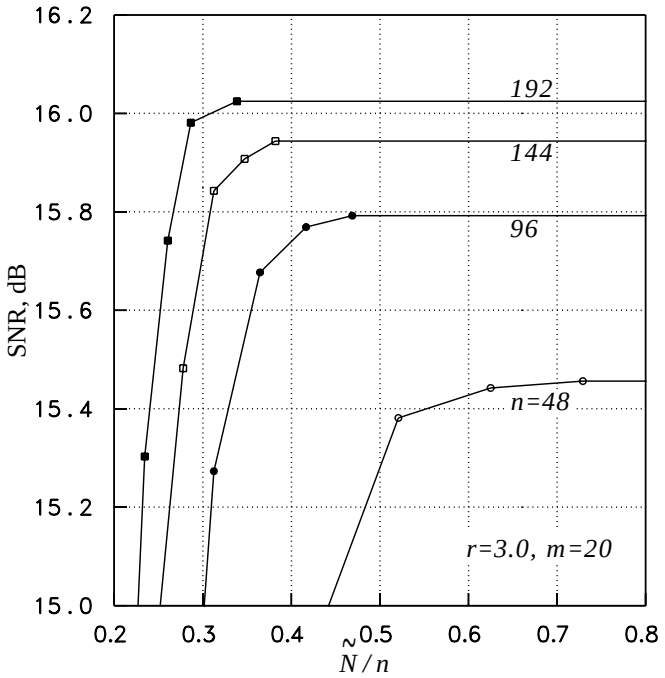

Figure 8: Performance of greedy search for an IID Gaussian source.

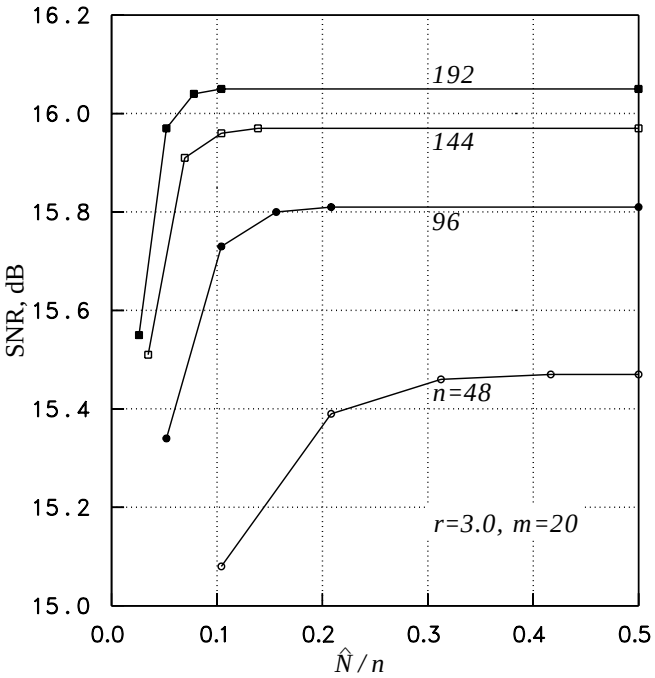

Figure 9: Performance of the LM-based search for an IID Gaussian source.

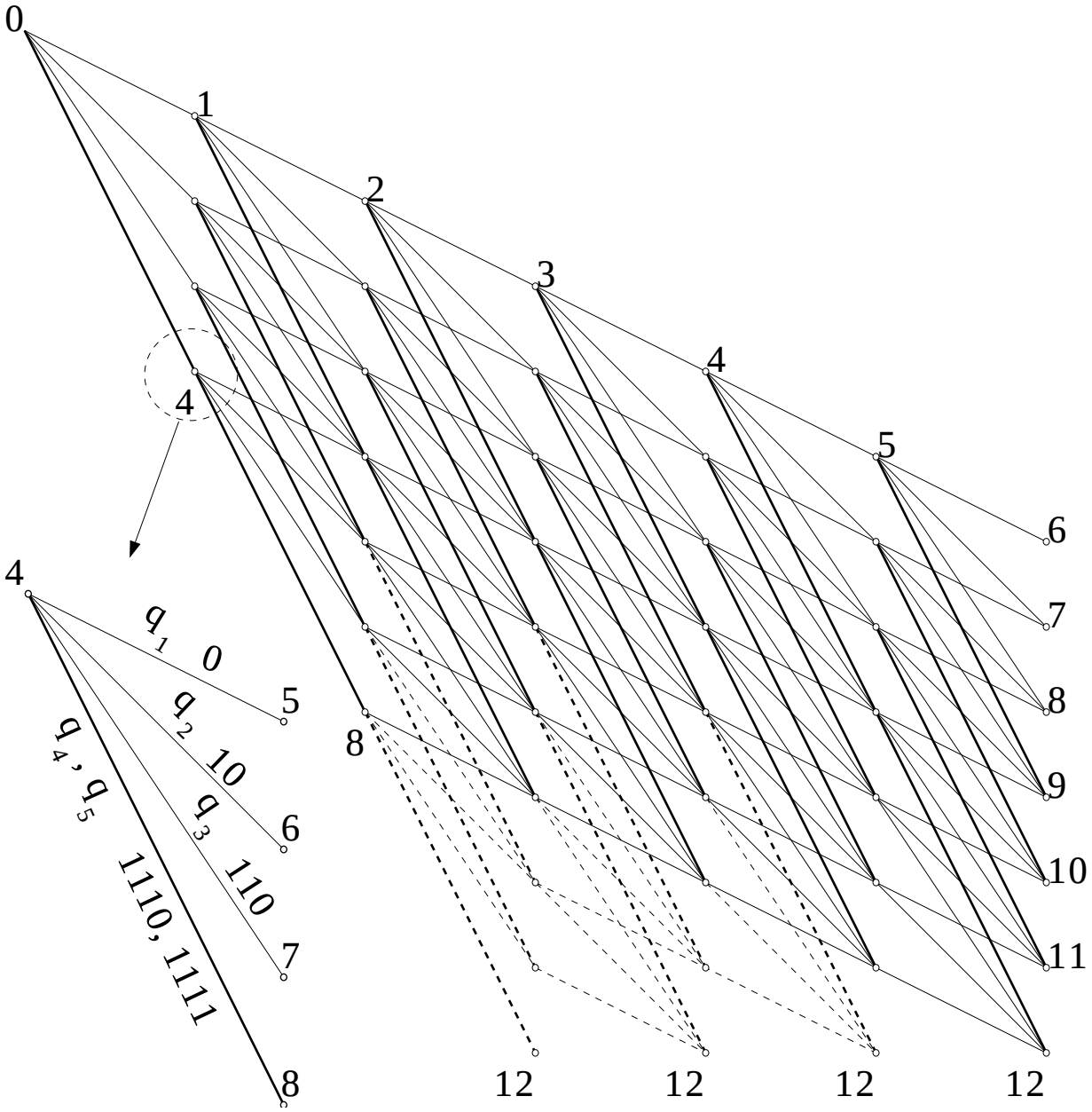

Figure 10: BCQ trellis example.

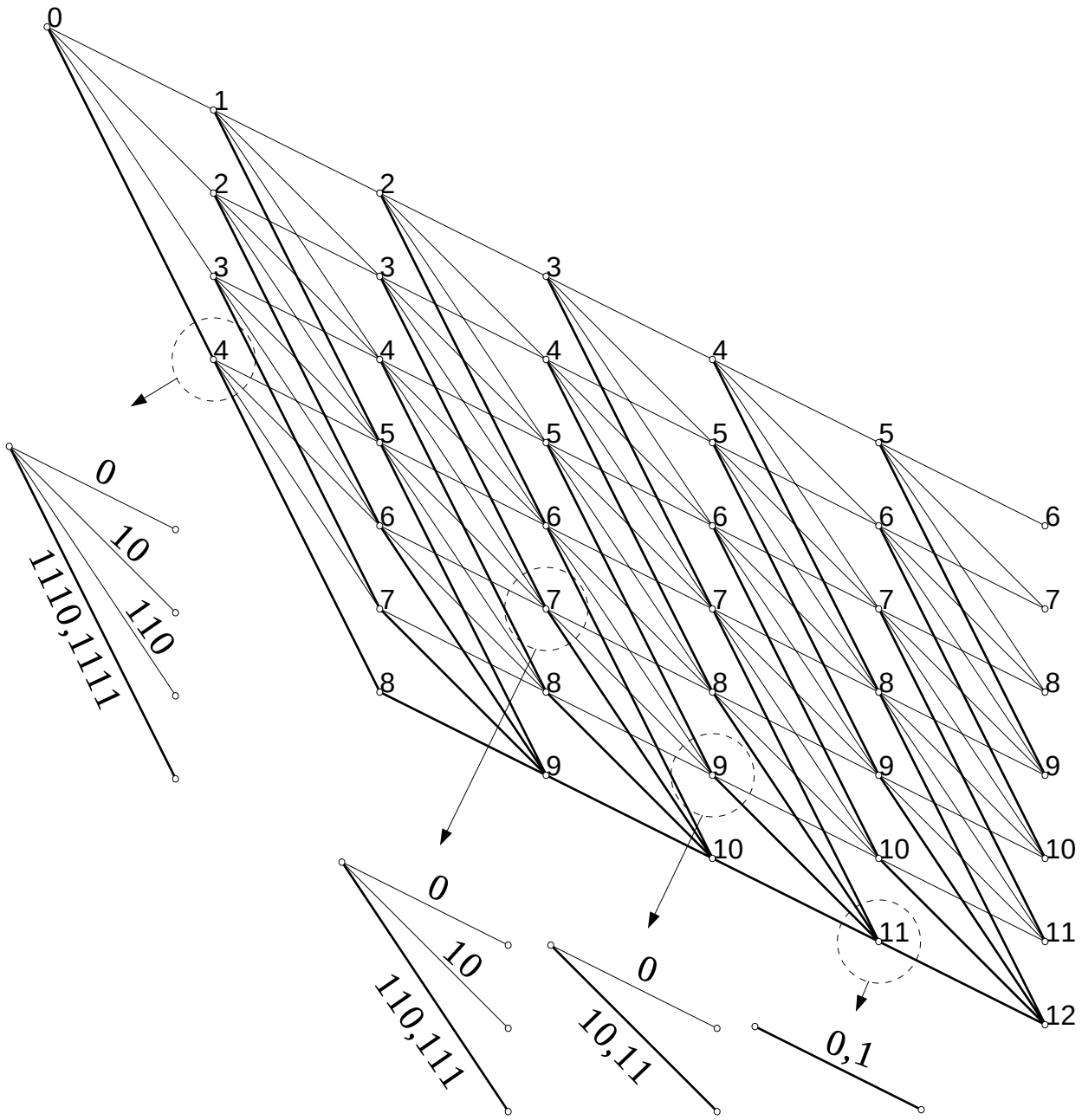

Figure 11: A node-varying BCQ trellis.