

# Quantization as Histogram Segmentation: Globally Optimal Scalar Quantizer Design in Network Systems \*

Dan Muresan  
Dept. of Electrical Engineering  
Stanford University  
muresan@stanford.edu

Michelle Effros  
Dept. of Electrical Engineering  
California Institute of Technology  
effros@caltech.edu

## Abstract

We propose a polynomial-time algorithm for optimal scalar quantizer design on discrete-alphabet sources. Special cases of the proposed approach yield optimal design algorithms for fixed-rate and entropy-constrained scalar quantizers, multi-resolution scalar quantizers, multiple description scalar quantizers, and Wyner-Ziv scalar quantizers. The algorithm guarantees globally optimal solutions for fixed-rate and entropy-constrained scalar quantizers and constrained optima for the other coding scenarios. We derive the algorithm by demonstrating the connection between scalar quantization, histogram segmentation, and the shortest path problem in a certain directed acyclic graph.

## 1 Introduction

A quantizer consists of a set of reproduction values (*codewords*, which together form a *codebook*) and a mapping from the source alphabet to the codebook. The set of alphabet symbols that are mapped to a certain codeword is called the *codecell* associated with that codeword. The set of all codecells forms a partition of the source alphabet. When the source alphabet is a discrete subset of some continuous alphabet, we call the discrete codecells *codegroups* (to distinguish them from the continuous codecells associated with the continuous super-alphabet).

Quantizer design can start with the codewords or with the codecells or codegroups. Most design algorithms focus on codebook design and define the codecells and codegroups as functions of the codebook. We here explore the alternative approach.

Our central result is a low-complexity, optimal scalar quantizer design algorithm for finite-alphabet sources. This approach applies to a variety of fixed- and variable-rate source codes involving a single source and one or more decoders. Examples include scalar quantization (SQ), multi-resolution scalar quantization (MRSQ), multiple description scalar quantization (MDSQ), and Wyner-Ziv scalar quantization

---

\*This work was supported in part by the Lee Center for Advanced Networking at Caltech.

(WZSQ). In each case the algorithm yields a global optimum in a constrained minimization problem. In SQ, this constrained minimum is identical to the unconstrained minimum, yielding the globally optimal code. In MRSQ, MDSQ, and WZSQ, the constrained minimum may differ from the unconstrained minimum.

Previous work [1, 2, 3, 4] uses polynomial-time dynamic programming algorithms for globally optimal fixed-rate, single-encoder, single-decoder SQ design. Our results generalize earlier techniques to variable-rate quantizers, quantizers with more than one decoder, and quantizers with side information at the decoder. In particular, our results include optimal, polynomial-time design algorithms for fixed- and variable-rate SQ, MRSQ, and MDSQ design with and without decoder side information; for SQ without side information, the design algorithm guarantees a globally optimal code. We also clarify the connection between quantization, segmentation, and the shortest path problem in a certain directed acyclic graph.

While the discussion that follows assumes finite-alphabet sources, our algorithm yields optimal scalar quantizers for any finite source model, including the case where the empirical model for a continuous-alphabet source is derived from a finite training set. Further, our algorithm yields approximating solutions to the optimal scalar quantizer design problem for continuous alphabet sources. This is achieved by applying the finite-alphabet algorithm to a discretization of the continuous alphabet.

## 2 Problem Set-Up

Each code design aims to optimize the tradeoff between expected rates and distortions. Here the distortion measure  $d(x, \hat{x})$  is assumed to be non-negative and the expectation is taken relative either to a known pmf on a finite alphabet or to a finite training set on an arbitrary alphabet. Given this effective restriction to finite alphabets and the assumption that the alphabet is ordered, in the remainder of this work we refer to scalar source alphabet  $x_1 \cdots x_N$  containing  $N < \infty$  symbols by the symbol indices  $\{1, 2, \dots, N\}$ . The alphabet and pmf are written as  $1 \cdots N$  and  $p[1] \cdots p[N]$ .

Given an encoder or decoder, optimization of the remaining portion is straightforward. We focus on optimal encoder design. Designing an encoder for  $x_1 \cdots x_N$  is equivalent to designing one or more partitions on  $1 \cdots N$ , where each set in a partition describes a codegroup. An SQ encoder requires one partition;  $M$ -DSQ and  $M$ -RSQ encoders use  $M$  partitions, one for each description or resolution.

Quantizer  $Q$  has *contiguous* codecells if for each encoder partition  $\mathcal{P}$  of  $Q$  there exists an increasing sequence of natural numbers  $T = \{t_k\}_{k=0}^{|\mathcal{P}|}$  with  $t_0 = 0$ ,  $t_{|\mathcal{P}|} = N$ , and  $\mathcal{P} = \{t_0 + 1 \cdots t_1, t_1 + 1 \cdots t_2, \dots, t_{|\mathcal{P}|-1} + 1 \cdots t_{|\mathcal{P}|}\}$ . For any threshold sequence  $T = \{t_k\}_{k=0}^K$  with  $t_0 = 0$  and  $t_K = N$ , we define the *partition assembly function* as the bijective map  $A(T) = \{t_0 + 1 \cdots t_1, t_1 + 1 \cdots t_2, \dots, t_{K-1} + 1 \cdots t_K\}$ . Each  $t_k$  (with the exception of  $t_0 = 0$ ) is the index of the last member in a codegroup.

The quantizer design algorithm proposed here finds, in each coding scenario, the globally optimal quantizer *among all quantizers with contiguous codecells*. The resulting SQ, MRSQ, MDSQ, or WZSQ is a globally optimal code if and only if the best possible performance for that code type can be achieved with contiguous codecells.

Under broad assumptions on the distortion measure, design under the codecell contiguity requirement guarantees a globally optimal code in the SQ case; unfortunately, the optimal MRSQ, MDSQ, and WZSQ with contiguous codecells may achieve performances worse than those of the corresponding optimal codes with non-contiguous codecells [5].

We view the histogram  $p[1] \cdots p[N]$  as a *probability signal*. Given the contiguity assumption, each codegroup on the source alphabet corresponds to a segment of the probability signal. Since the performance of a partition – measured by the expected rate and distortion of the corresponding quantizer encoder and its optimal decoder – is a function of the probability signal, we treat the partition-design problem on alphabet  $1 \cdots N$  as a segmentation problem on signal  $p[1] \cdots p[N]$ . We want to design the segmentation that achieves the best possible tradeoff between expected rates and expected distortions, and we want the design algorithm to be fast.

We consider both true partition optimization relative to source alphabet  $1 \cdots N$  and approximate optimization where partitions are constrained to a coarse grid. We achieve the coarse grid by dividing the symbols of  $1 \cdots N$  into  $\hat{N}$  contiguous, non-overlapping cells  $\{G_k\}_{k=1}^{\hat{N}}$  that partition  $1 \cdots N$ . Given the coarse grid  $\mathcal{G} = \{G_k\}_{k=1}^{\hat{N}}$ , the partition design problem on  $1 \cdots N$  simplifies to a partition design problem on a fixed, ordered set of indivisible cells. While there are  $2^{N-1}$  partitions of the original alphabet, only  $2^{\hat{N}-1}$  of these are compatible with the  $\hat{N}$ -cell grid. Guaranteeing true optimality requires  $\hat{N} = N$ , but using  $\hat{N} < N$  allows for faster optimization.

### 3 Scalar Quantization

This section treats variable-rate SQ design; fixed-rate SQ design appears in [1, 2, 3, 4].

Let  $D(\mathcal{P})$  and  $R(\mathcal{P})$  be the expected distortion and output entropy, respectively, for the variable-rate SQ described by partition  $\mathcal{P}$ . For any  $\mathcal{P}$  with  $(D(\mathcal{P}), R(\mathcal{P}))$  on the lower convex hull of the first-order operational distortion-rate function, there exists a Lagrangian multiplier  $\lambda > 0$  for which  $\mathcal{P}$  minimizes  $J(\mathcal{P}, \lambda) = D(\mathcal{P}) + \lambda R(\mathcal{P})$  over all partitions. We therefore use the Lagrangian  $J(\mathcal{P}, \lambda)$  as the optimization criterion to design an entropy-constrained scalar quantizer (ECSQ) partition.

Note that rate and distortion (and thus the Lagrangian) are additive over codegroups. For a partition  $\mathcal{P}$ ,  $R(\mathcal{P}) = \sum_{C \in \mathcal{P}} r(C)$  where  $r(C) = -p(C) \log p(C)$  and  $p(C) = \sum_{n \in C} p[n]$  are the *partial rate* and codegroup probability.<sup>1</sup> Similarly,  $D(\mathcal{P}) = \sum_{C \in \mathcal{P}} d(C)$ , where  $d(C) = \sum_{n \in C} p[n] d(x_n, \mu(C))$  and  $\mu(C)$  are the *partial distortion* and optimal codeword for  $C$ . When  $d(x, \hat{x}) = (x - \hat{x})^2$ ,  $\mu(C) = \sum_{n \in C} p[n] x_n / p(C)$ .

Signal segmentation and the single-source shortest path problem in a weighted directed acyclic graph (WDAG) are related, and others have exploited this observation to obtain rate-distortion optimal encoders in codes that rely on segmentation. In this paper we again use the relationship between the segmentation and shortest path problems to find a rate-distortion-optimal segmentation. Here we segment the probability signal during code design rather than segmenting the source signal during encoding. Our segmentation equals  $\mathcal{P}$ , and  $\mathcal{P}$  is optimal when it minimizes  $J(\mathcal{P}, \lambda)$ .

<sup>1</sup>All logarithms in this work are base-2.

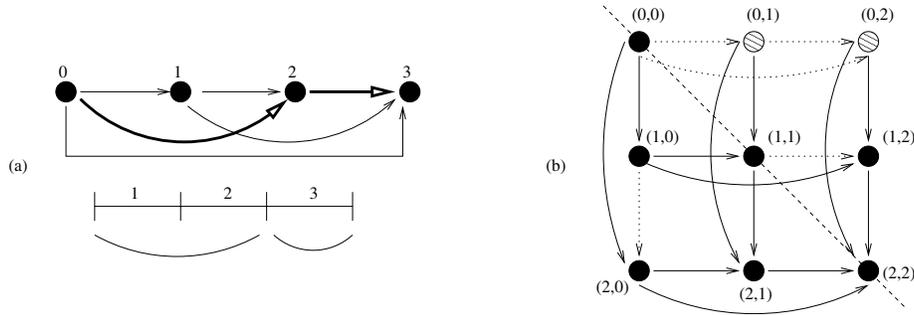


Figure 1: Partial RD graphs. (a) SQ ( $\hat{N} = 3$ ); the path describing the partition  $\{1 \cdots 2, 3 \cdots 3\}$  appears in bold. (b) 2DSQ ( $\hat{N} = 2$ ).

Our optimal segmentation algorithm is equivalent to the shortest-path algorithm on a WDAG called the *partial RD graph*. The partial RD graph for ECSQ has  $\hat{N} + 1$  vertices corresponding to the  $\hat{N} + 1$  intervals of a line delimited by  $\hat{N}$  points; the points are a topological abstraction of the  $\hat{N}$  alphabet cells and must satisfy the total ordering induced by the original alphabet (for cells  $c_1$  and  $c_2$ ,  $c_1 < c_2$  if and only if  $x < y$  for all  $x \in c_1, y \in c_2$ ). For every pair  $(u, v)$  with  $u < v$  there is an *arc* (directed edge) from  $u$  to  $v$ . The arc corresponds to a codegroup  $C_{u+1 \cdots v}$  comprising cells  $u + 1 \cdots v$ , and the weight is the Lagrangian cost  $d(C_{u+1 \cdots v}) + \lambda r(C_{u+1 \cdots v})$  for that codegroup. The graph has  $\hat{N}(\hat{N} + 1)/2$  arcs (see Fig. 1(a)).

Let  $S_u$  denote the set of partitions on cells  $1 \cdots u$  ( $S_0 = \emptyset$ ). For any  $1 \leq v \leq \hat{N}$ ,  $S_v$  can be obtained by extending “shorter” partitions with the appropriate codegroup  $S_v = \cup_{u=0}^{v-1} \cup_{\mathcal{P} \in S_u} \mathcal{P} \cup \{C_{u+1 \cdots v}\}$ . All partitions  $\mathcal{P} \in S_u$  must be extended with the same codegroup  $C_{u+1 \cdots v}$  to obtain partitions in  $S_v$ . In the partial RD graph, then, each vertex  $v$  corresponds to the partition set  $S_v$ , and each arc corresponds to a codegroup extending a collection of “shorter” partitions to a collection of “longer” partitions. A path from 0 to some vertex  $v$  shows how to “grow” a unique partition in  $S_v$  by successively adding codegroups. Each path from 0 to  $\hat{N}$  describes a unique partition.

To prove that optimal SQ design is equivalent to finding the shortest path between vertices 0 and  $\hat{N}$ , we must first show a one-to-one correspondence between partitions and paths in the graph. A path of length  $l$  from 0 to  $\hat{N}$  is a chain of edges  $(v_0, v_1), (v_1, v_2), \dots, (v_{l-1}, v_l)$  with  $v_0 = 0$  and  $v_l = \hat{N}$ . By design,  $(v_i, v_{i+1})$  is an arc if and only if  $v_i < v_{i+1}$ . Therefore, we can associate bijectively to each arc  $(v_1, v_2)$  the codegroup  $v_1 + 1 \cdots v_2$ . For a partition, consecutive codegroups must be adjacent; for a path, consecutive arcs must share one vertex. Therefore an enumeration of the arcs corresponding to the codegroups in a partition forms a path, and vice-versa. This establishes the required two-way correspondence. Since the total weight on a path from 0 to  $\hat{N}$  is equal (by construction) to the Lagrangian for the partition corresponding to the path, we have the desired proof of correctness.

The single-source shortest path problem in a WDAG with  $V$  vertices and  $E$  edges can be solved in  $O(V + E)$  time using dynamic programming. The first step is to sort the vertex set topologically, that is, to order the  $V$  vertices of the graph in a

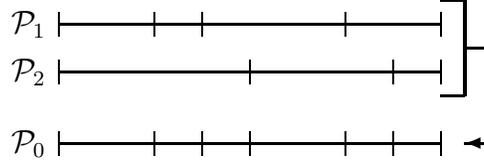


Figure 2: Side partitions  $\mathcal{P}_1$  and  $\mathcal{P}_2$  and central partition  $\mathcal{P}_0$

sequence  $\{v_k\}_{k=1}^V$  such that for  $k > \ell$ , there are no arcs from  $v_k$  to  $v_\ell$  (this is always possible in an acyclic graph). In the partial RD graph used for optimal ECSQ design, the vertex labels give the topological order of the vertices. In general, ordering the vertices requires  $O(V + E)$  steps [5].

Given the sorted vertex sequence  $\{v_k\}_{k=1}^V$ , the dynamic programming algorithm proceeds as follows. At each step  $\ell \in 1 \cdots V$ , compute the shortest path to vertex  $v_\ell$  as  $s[\ell] = \min_k (s[k] + w_{k,\ell})$ , where  $w_{k,\ell}$  is the weight on arc  $(v_k, v_\ell)$  and the minimization is performed over the *ancestor set* of vertex  $v_\ell$  (i.e., the set of vertices  $v_k$  that have outgoing arcs  $(v_k, v_\ell)$ ). Since the sequence  $\{v_k\}_{k=1}^V$  is sorted topologically, the ancestors of any vertex occur before it in the sequence, and the shortest paths  $s[k]$  on the right hand side are well defined (set  $s[k] = 0$  if  $v_k$  has no ancestors).

The partial RD graph has  $O(\hat{N}^2)$  edges. Therefore, given the weights, optimal SQ design can be performed in  $O(\hat{N}^2)$  time. Since finding each of the  $O(\hat{N}^2)$  weights requires  $O(N)$  operations in general, the preprocessing required to obtain the graph structure takes  $O(\hat{N}^2 N)$  time. The preprocessing cost can be reduced to  $O(N)$  when  $d(x, \hat{x}) = (x - \hat{x})^2$  [3] and  $O(N^2)$  when  $d(x, \hat{x})$  is monotonic in  $|x - \hat{x}|$  [4]. Performing the weight calculations during the shortest path algorithm (rather than in an independent preprocessing step) yields a total complexity for globally optimal ECSQ design of  $O(\hat{N}^2 + N)$ ,  $O(N^2)$ , or  $O(\hat{N}^2 N)$ , depending on the distortion measure.

## 4 Multiple-Decoder Systems

We next generalize the approach described above to single-source, multiple-decoder quantizers like MRSQ and MDSQ. MRSQ is scalar quantization yielding embedded source descriptions that can be decoded in part or in whole, with an expected reproduction quality that improves as a function of the rate of the decoded sequence. MDSQ is scalar quantization using a packetized source description, with required reconstruction under any combination of received and lost packets.

We focus our description on variable-rate MDSQ design with  $M = 2$  (i.e., 2DSQ) for simplicity. The generalizations to fixed-rate coding,  $M > 2$ , MRSQ, and other examples of MDSQs with restricted packet-loss scenarios are extensions of the variable-rate  $M = 2$  solution, as described briefly at the end of this work and in full in [5].

The encoder of a 2DSQ gives two separate data descriptions and is therefore defined by two distinct partitions,  $\mathcal{P}_1$  and  $\mathcal{P}_2$ , of the alphabet. The decoder may receive either of the two descriptions alone, in which case the problem reduces to ordinary SQ with partition  $\mathcal{P}_1$  or  $\mathcal{P}_2$ . More interestingly, both descriptions may be received; in this case, the decoder knows that the source symbol  $x$  satisfies  $x \in C_1 \cap C_2$

for some codegroups  $C_1 \in \mathcal{P}_1$  and  $C_2 \in \mathcal{P}_2$ , yielding an effective underlying partition  $\mathcal{P}_0 = \{C_1 \cap C_2 : C_1 \in \mathcal{P}_1, C_2 \in \mathcal{P}_2\}$  of the data. Partition  $\mathcal{P}_0$  is called the *central partition*, while  $\mathcal{P}_1$  and  $\mathcal{P}_2$  are called *side partitions* (see Fig. 2). The intersection needed to compute the central partition involves  $O(|\mathcal{P}_1| + |\mathcal{P}_2|)$  operations since most pairs of codegroups don't intersect. A plausible algorithm that does not explicitly perform codegroup intersections is to merge-sort the threshold sequences of the side partitions as  $\mathcal{P}_0 = A(\text{MERGE}(A^{-1}(\mathcal{P}_1), A^{-1}(\mathcal{P}_2)))$ . (Recall that  $A$  is the mapping from each threshold sequence to its corresponding partition.)

Using the notation from Section 3, let  $R_i = R(\mathcal{P}_i)$  ( $i \in 1 \cdots 2$ ) and  $D_i = D(\mathcal{P}_i)$  ( $i \in 0 \cdots 2$ ). Then  $(R_i, D_i)$  ( $i \in 1 \cdots 2$ ) is the expected rate-distortion performance when only the  $i$ th description is received, and  $D_0$  is the expected distortion when both descriptions are received. (Rate  $R(\mathcal{P}_0)$  is not used since partition  $\mathcal{P}_0$  is described only through the description of  $\mathcal{P}_1$  and  $\mathcal{P}_2$ .) Again  $D(\mathcal{P}_i) = \sum_{C \in \mathcal{P}_i} d(C)$  and  $R(\mathcal{P}_i) = \sum_{C \in \mathcal{P}_i} r(C)$ , with  $d(C) = \sum_{n \in C} p[n]d(x_n, \mu(C))$  and  $r(C) = -p(C) \log p(C)$ .

Let  $\mathcal{D}^{\text{vr}} = \{(R(\mathcal{P}_i)|_{i=1}^2, D(\mathcal{P}_i)|_{i=0}^2) : \mathcal{P}_1 \text{ and } \mathcal{P}_2 \text{ induce } \mathcal{P}_0\}$  be the set of rate-distortion vectors achievable through variable-rate MDSQ. Any pair of partitions  $(\mathcal{P}_1, \mathcal{P}_2)$  achieving a point  $(R_1, R_2, D_0, D_1, D_2)$  on the lower boundary of  $\mathcal{D}^{\text{vr}}$  is in some sense optimal. For any  $(\mathcal{P}_1, \mathcal{P}_2)$  with  $(R(\mathcal{P}_i)|_{i=1}^2, D(\mathcal{P}_i)|_{i=0}^2)$  on the lower convex hull of  $\mathcal{D}^{\text{vr}}$ , there exist non-negative Lagrangian vectors  $\lambda^2$  and  $\nu^2$  for which  $(\mathcal{P}_1, \mathcal{P}_2)$  minimizes  $J^{\text{vr}}((\mathcal{P}_1, \mathcal{P}_2), \nu^2, \lambda^2) = D_0 + \sum_{i=1}^2 [\nu_i D(\mathcal{P}_i) + \lambda_i R(\mathcal{P}_i)]$ . Therefore, we use  $J^{\text{vr}}((\mathcal{P}_1, \mathcal{P}_2), \nu^2, \lambda^2)$  as the partition optimization criterion in variable-rate 2DSQs.

In 2DSQ design, we *jointly* optimize the two side partitions for the encoder. Since the  $D_0$  term relies on the central partition, we must keep track of the central partition (which is a dependent function of the 2 side partitions) throughout the algorithm.

We generalize the partial RD graph using vertex labels  $(v_1, v_2)$ , where  $(v_1, v_2)$  represents a set of *partition pairs*  $(\mathcal{P}_1, \mathcal{P}_2)$  such that  $\mathcal{P}_1$  spans  $0 \cdots v_1$  and  $\mathcal{P}_2$  spans  $0 \cdots v_2$  (we ignore the central partition for the time being).

Construction of the edge set presents a certain difficulty; naively, there should be an arc from vertex  $(v_1, v_2)$  to all vertices  $(v'_1, v_2)$  and  $(v_1, v'_2)$  such that  $v'_1 > v_1$  and  $v'_2 > v_2$ , since we can extend a partition pair by adding a codegroup to either of the 2 side partitions, as shown in the example in Fig. 1(b). (In the discussion that follows, we remove the shaded vertices and dotted arcs, but for now they should be treated like their solid counterparts.) We emphasize the fact that only one side partition may be extended at a time, thus only one side codegroup is added by each arc. The difficulty arises when we attempt to assign weights to the edges. As we next demonstrate, some partitions must not be represented in their associated vertex, and to guarantee that exclusion, some edges must be removed from the graph.

First, we must account for the central partition. For vertices  $(v, v)$  (where the 2 side partitions have equal length),  $\mathcal{P}_0$  can be computed as explained above and illustrated in Fig. 2. When the side partitions have different lengths, however, the central partition can be determined only up to the end of the shorter side partition. Consider the  $(v_1, v_2)$  partition pair with  $v_1 > v_2$  shown in Fig. 3(a). Naively, we would find all intersections of side codegroups, so that the last central codegroup would be  $v_2 \cdots v_1$ ; however, adding a new side codegroup  $v_2 \cdots v'_2$  to  $\mathcal{P}_2$ , as illustrated in the lower portion of Fig. 3(a), breaks the central codegroup  $v_2 \cdots v_1$  into two new central

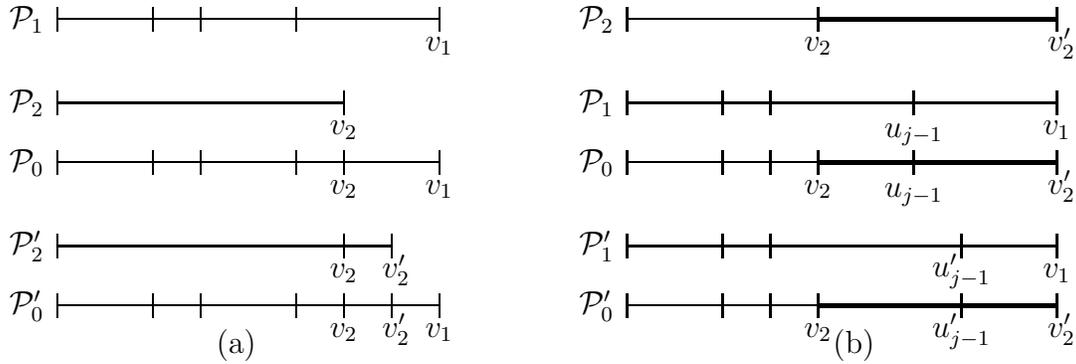


Figure 3: (a) The central partition is determined only to the end of the shorter side partition. When  $v_2 \cdots v'_2$  is added to  $\mathcal{P}_2$ , the central partition changes in  $v_2 \cdots v_1$  but does not change on values less than  $\min\{v_1, v_2\}$ . (b) The shorter side partition is too short. Here,  $v_2 < u_{j-1}$  implies that the central partition on  $v_2 \cdots v_1$  varies with  $u_{j-1}$ .

codegroups  $v_2 \cdots v'_2$  and  $v'_2 \cdots v_1$  that depend on  $v'_2$ . While the central codegroups lying within the range  $1 \cdots \min(v_1, v_2)$  are not influenced by the subsequent addition of side codegroups, the central codegroups in  $\min(v_1, v_2) \cdots \max(v_1, v_2)$  may vary as a function of future additions to the shorter side partition.

The consequence of ambiguity in the central partition definition is a corresponding difficulty in assigning weights to the graph's arcs. Each arc corresponds to a single codegroup being added to one of the side partitions and one or possibly multiple codegroups being added to the central partition. To be able to assign a unique weight to that arc, the total Lagrangian cost of the newly added codegroups must be the same regardless of the "internal structure" of the partition being extended. That is, the cost must be a function only of the labels of the two vertices that it joins. The codegroup added to one of the side partitions is clearly determined by the two vertex labels only, but the central codegroups are not, in general. For example, consider the situation shown in Fig. 3(b). Here we extend the shorter partition  $\mathcal{P}_2$  by adding group  $v_2 \cdots v'_2$ . The figure shows two examples of possible values for partition  $\mathcal{P}_1$ . Both examples correspond to a single path from node  $(v_1, v_2)$  to node  $(v_1, v'_2)$ . Yet the resulting central codegroups, and thus the Lagrangian costs, differ in the two cases, making the label for the arc ambiguous. This problem seems to arise from the fact that  $\mathcal{P}_2$  is "too short," one and a half groups shorter than  $\mathcal{P}_1$ . We need a way to keep side partitions more "even" to avoid this type of problem.

The *codegroup lag*, defined using the two threshold sequences  $\{t_k\}_{k=0}^J = A^{-1}(\mathcal{P}_1)$  and  $\{u_k\}_{k=0}^K = A^{-1}(\mathcal{P}_2)$ , provides a measure of how "even" two partitions are; partitions must be aligned on the left side ( $u_0 = v_0$ ) for the following definition to apply. We say that  $\mathcal{P}_1$  lags  $\mathcal{P}_2$  by  $L$  codegroups if there are  $L$  codegroups in  $\mathcal{P}_2$  not lying entirely inside the range covered by  $\mathcal{P}_1$  (i.e.,  $u_{K-L} \leq t_J = v_1 < u_{K-L+1}$ ). We want to keep the codegroup lag between the two side partitions at most 1. (A lag of zero means that the partitions are of equal total length, a case which we handle later in this section.) Call a partition pair with codegroup lag less than or equal to 1 *valid*.

Suppose that  $\mathcal{P}_2$  is the shorter partition ( $v_2 < v_1$ ). Then a lag of 1 implies that the range  $v_2 \cdots v_1$  is included in the last codegroup of  $\mathcal{P}_1$  (i.e.,  $t_{J-1} < u_K = v_2 < t_J = v_1$ ).

Adding codegroup  $v_2 \cdots v'_2$  to  $\mathcal{P}_2$  results in a single central codegroup  $v_2 \cdots \min(v_1, v'_2)$  being generated, regardless of the structure of the side partitions; therefore we achieve our objective. The weight on each arc is defined as the Lagrangian cost associated with one new side codegroup and one new central codegroup.

We now have a condition (validity) that guarantees that adding side codegroups results in predictable central codegroups being generated; we next constrain the partial RD graph to obey that condition. We want to keep partition pairs with codegroup lag more than 1 unrepresented in the vertices of the graph. Since the partition pairs represented in each vertex are defined by the paths between  $(0, 0)$  and that vertex, it is enough to break certain malignant paths by removing arcs from the graph.

We accomplish our goal by eliminating arcs that extend the longer side partition in a partition pair. Extending the longer side creates a lag of 2 or more; extending the shorter side partition does not increase lag when  $v_1 \neq v_2$  (the exact change in lag depends on the exact structure of the partitions), and changes lag from 0 to 1 when  $v_1 = v_2$ . By an inductive argument, extending the shorter partition keeps lag at 1 or 0. When  $v_1 = v_2$ , no central codegroups are added regardless of which side partition is extended and by how much; to avoid duplicate paths, we extend  $\mathcal{P}_1$  if  $v_1 = v_2$ .

Our new *partition extension rule* allows only two types of arcs:  $((v_1, v_2), (v_1, v'_2))$  with  $v_1 > v_2, v'_2 > v_2$  and  $((v_1, v_2), (v'_1, v_2))$  with  $v_1 \leq v_2, v'_1 > v_1$ . In the example of a partial RD graph for 2DSQ given in Fig. 1(b), the disallowed arcs are dotted. When we eliminate these arcs, some vertices become inaccessible from  $(0, 0)$  and therefore cannot be visited; we eliminate these vertices and the arcs originating from them. (The inaccessible vertices are  $\{(0, v_2) : v_2 > 0\}$ ; they are shaded in Fig. 1(b).)

We are left with a graph of  $(\hat{N} + 1)^2$  vertices ( $\hat{N}$  of which are inaccessible) and  $\hat{N}(\hat{N} + 1)(4\hat{N} + 5)/6$  arcs ( $\hat{N}^2$  of which originate in inaccessible vertices).

The weight  $W$  on an arc  $((v_1, v_2), (v'_1, v'_2))$  is as follows:

- When  $v_1 = v_2 = v'_2 = v$ ,  $C_1 = C_{v \dots v'_1}$  is added to  $\mathcal{P}_1$ ;  $W = \nu_1 d(C_1) + \lambda_1 r(C_1)$ .
- When  $v_2 = v'_2 > v_1$ ,  $C_1 = C_{v_1 \dots v'_1}$  is added to  $\mathcal{P}_1$  and  $C_0 = C_{v_1 \dots \min(v'_1, v_2)}$  is generated in  $\mathcal{P}_0$ ;  $W = d(C_0) + \nu_1 d(C_1) + \lambda_1 r(C_1)$ .
- When  $v_1 = v'_1 > v_2$ ,  $C_2 = C_{v_2 \dots v'_2}$  is added to  $\mathcal{P}_2$  and  $C_0 = C_{v_2 \dots \min(v_1, v'_2)}$  is generated in  $\mathcal{P}_0$ ;  $W = d(C_0) + \nu_2 d(C_2) + \lambda_2 r(C_2)$ .

When  $d(x, \hat{x}) = (x - \hat{x})^2$  each weight can be computed in constant time using [3] since at most two codegroups are generated by adding an arc to a given path.

Given the 2DSQ partial RD graph, running a shortest path algorithm with source  $(0, 0)$  and sink  $(\hat{N}, \hat{N})$  yields the optimal 2DSQ with contiguous codecells.

To prove the algorithm's correctness, we exhibit a bijection between paths from  $(0, 0)$  to  $(\hat{N}, \hat{N})$  and partition pairs. One direction is obvious: paths still describe how to "grow" partition pairs by successive extensions, and we only need to keep track of  $\mathcal{P}_1$  and  $\mathcal{P}_2$  for this part of the proof. It is sufficient to map each arc to the corresponding codegroup being added to either  $\mathcal{P}_1$  or  $\mathcal{P}_2$ , (based on the labels of the origin and destination vertices of the arc) to obtain the partition pair associated to a path.

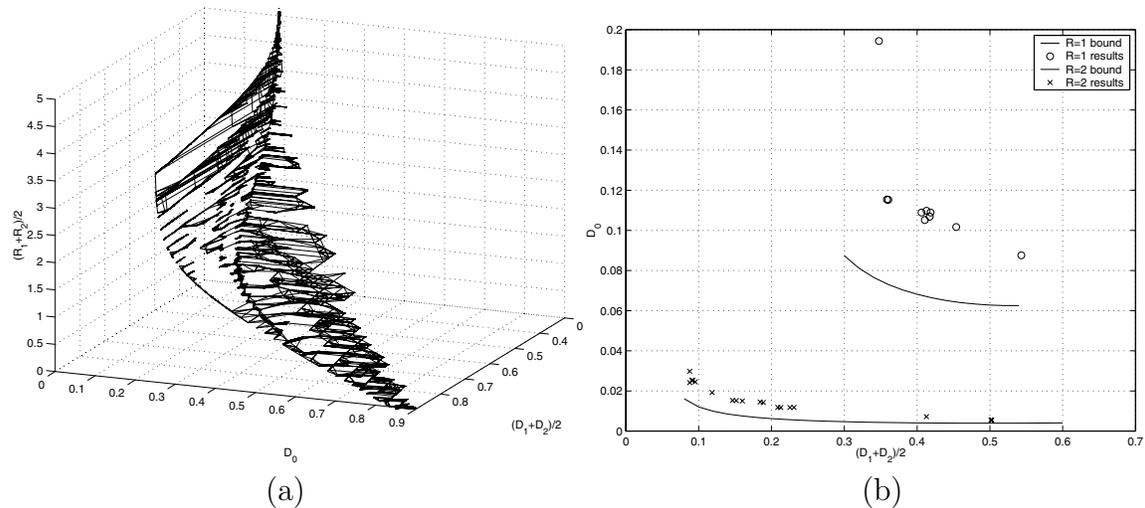


Figure 4: 2DSQ on a truncated Gaussian ( $\hat{N} = 48$  segments, each of width  $1/6$ ).

Showing that there is a unique path for each pair of partitions on  $\hat{N}$  requires some new terminology. Given partitions  $\mathcal{P}$  and  $\mathcal{P}'$  we say that  $\mathcal{P}$  is a *truncated version* of  $\mathcal{P}'$  if  $\mathcal{P}' \subseteq \mathcal{P}$  and the two partitions are aligned on the left side (i.e., the threshold sequences  $\{t_i\}_{i=0}^l$  and  $\{t'_i\}_{i=0}^{l'}$  must satisfy  $l' \leq l$ ,  $t_i = t'_i$  for  $i \in 0 \cdots l'$ ), and  $t_0 = t'_0 = 0$ . Similarly, we define  $(\mathcal{P}'_1, \mathcal{P}'_2)$  to be a truncated version of  $(\mathcal{P}_1, \mathcal{P}_2)$  if  $\mathcal{P}'_1$  and  $\mathcal{P}'_2$  are truncated versions of  $\mathcal{P}_1$  and  $\mathcal{P}_2$ , respectively. (Since the partitions in each pair are aligned on the left side, all four partitions are aligned on the left side.) We say that codegroup  $C_i$  added to  $\mathcal{P}'_i$  ( $i \in 1 \cdots 2$ ) extends  $(\mathcal{P}'_1, \mathcal{P}'_2)$  according to  $(\mathcal{P}_1, \mathcal{P}_2)$  if the extension is a truncated version of  $(\mathcal{P}_1, \mathcal{P}_2)$ . If  $(\mathcal{P}'_1, \mathcal{P}'_2)$  is not a truncated version of  $(\mathcal{P}_1, \mathcal{P}_2)$ , none of its extensions are either.

Now consider an arbitrary valid partition pair  $(\mathcal{P}_1, \mathcal{P}_2)$ , and let  $L$  be the sum of the number of codegroups in  $\mathcal{P}_1$  and the number of codegroups in  $\mathcal{P}_2$ . We next show by induction that for any  $l \in 0 \cdots L$  the graph contains a unique path of length  $l$  that begins at  $(0, 0)$  and describes a truncated version of  $(\mathcal{P}_1, \mathcal{P}_2)$ . For  $l = 0$ , this is immediate. For  $l > 0$ , any path of length  $l$  can be decomposed into a path of length  $l - 1$  and an extra arc. If the length  $l$  path is to correspond to a truncated version of  $(\mathcal{P}_1, \mathcal{P}_2)$ , we have seen that the length  $l - 1$  path must also. There exists only one path of length  $l - 1$  that satisfies this condition, and the extension rule guarantees that there exists a unique arc that extends it according to  $(\mathcal{P}_1, \mathcal{P}_2)$ . (The latter statement uses the fact that both  $(\mathcal{P}_1, \mathcal{P}_2)$  and the partition pair corresponding to the length- $(l - 1)$  path are valid.) Therefore the set of paths of length  $l$  satisfying the condition has cardinality exactly one.

Fig. 4 demonstrates a simple application of the 2DSQ design algorithm. The source pdf is a unit-variance Gaussian. The Lagrangian multipliers are forced to be symmetric ( $\lambda_1 = \lambda_2$ ,  $\nu_1 = \nu_2$ ). Under these circumstances, the following symmetry property holds: if  $(R_1, R_2, D_0, D_1, D_2)$  is a solution of the unconstrained problem, then  $(R_2, R_1, D_0, D_2, D_1)$  also is (for the same multiplier values). In Fig. 4(a), we plot  $(D_0, (D_1 + D_2)/2, (R_1 + R_2)/2)$  for various multiplier values. The second plot details a “slice” of the multi-dimensional RD surface for two fixed values of  $R$  (1

and 2 bits per channel), along with the theoretical rate-distortion bounds for the two cases. Note that the Gaussian pdf was approximated by  $\hat{N} = 48$  histogram segments, each of width  $\frac{1}{6}$  (our method cannot deal with continuous distributions — they must be discretized first); therefore, our results are approximate and include an irreducible error floor which could be made arbitrarily low by increasing the number of segments.

## 5 Extensions

Due to space considerations, we here exclude a variety of algorithm extensions described in [5]. Those extensions include fixed- and variable-rate  $M$ -DSQs for  $M \geq 2$ . These codes require an increase in design complexity, but complexity in both cases remains polynomial. In fixed-rate  $M$ -DSQ design, the complexity increase arises from the need to constrain the partition sizes, which requires inclusion of the current partition size in the vertex labels. In  $M$ -DSQ with  $M > 2$ , the complexity increases from the  $M$ -dimensional vertex labels (corresponding to the  $M$  side partitions) and from the need to track  $2^M - M - 1$  non-trivial induced *intersection partitions* (the analogues of the central partition from 2DSQ).

The above MDSQ discussion assumes that all packet loss scenarios occur with non-zero probability. A variety of codes, including MRSQ, can be described as MDSQs with restricted allowed packet loss scenarios. In [5], we modify the general MDSQ design algorithm for use with arbitrary scenario sets. The discussion treats both an optimized design algorithm for the highly structured packet-loss framework of MRSQ and a general algorithm for arbitrary restricted packet-loss environments. In both cases, the modifications increase computational complexity, but not beyond the polynomial landmark (in terms of alphabet symbols).

The final extension described in [5] treats the problem of fixed- and variable-rate SQ, MDSQ, and restricted MDSQ design when side information is available to the decoder but not to the encoder. The design proceeds along lines similar to those discussed above; the result is a family of optimal Wyner-Ziv scalar quantizers.

## References

- [1] J. D. Bruce. *Optimum Quantization*. PhD thesis, M.I.T., Cambridge, May 1964.
- [2] D. K. Sharma. Design of absolutely optimal quantizers for a wide class of distortion measures. *IEEE Trans. on Information Theory*, IT-24(6):693–702, 1978.
- [3] X. Wu. *Algorithmic approach to mean-square quantization*. PhD thesis, University of Calgary, 1988.
- [4] X. Wu and K. Zhang. Quantizer monotonicities and globally optimal scalar quantizer design. *IEEE Trans. on Information Theory*, IT-39(3):1049–1053, 1993.
- [5] D. Muresan and M. Effros. Quantization as histogram segmentation: globally optimal scalar quantizer design in network systems. 2001. In preparation.