A NEW ALGORITHM FOR OPTIMUM BIT LOADING IN SUBBAND CODING

Manish Vemulapalli1, Soura Dasgupta1, and Ashish Pandharipande2

1Department of Electrical & Computer Engineering
The University of Iowa
Iowa City, IA-52242, USA.
mvemulap, dasgupta@engineering.uiowa.edu

2Department of Electrical & Computer Engineering
University of Florida
Gainesville, FL 32611-6130, USA.
ashish dsp.ufl.edu

ABSTRACT

In this paper we present an efficient bitloading algorithm for subband coding. The goal is to effect an optimal distribution of positive integer bit values among various subchannels to achieve a minimum distortion error variance. Existing algorithms in the literature grow with the total number of bits that must be distributed. The novelty of our algorithm lies in the fact that its complexity is independent of the total number of bits to be allocated.

1. INTRODUCTION

An important problem in subband coding is bitloading. Specifically, for an $N$-subchannel system in this problem is a special case of the more general problem of finding $b_k$ to

$$\min \phi_k(b_k) = \sum_{k=1}^{N} \phi_k(b_k)$$

subject to

$$\sum_{k=1}^{N} b_k = B, b_k \in \{0, 1, ... B\}$$

where $\phi_k$ is a convex function, and $B$ is a positive integer. In subband coding

$$\phi_k(b_k) = \alpha_k 2^{-2b_k}$$

where $\alpha_k$ is determined by the signal variance in the $k$-th subchannel, and $b_k$ is assigned to the $k$-th subchannel. Further $\alpha_k$ increases with increasing signal variance.

It is recognized that for general convex functions $\phi_k(\cdot)$, the above constrained minimization grows in complexity with the size of $B$. Since $B$ can be large, it is important to formulate algorithms for which the complexity bound is independent of $B$.

$$\phi_k(b_k) = \alpha_k 2^{-b_k}$$

To place this work in context we note the presence of several bit loading algorithms in the literature but mostly from the communications perspective where

$$\phi_k(b_k) = \alpha_k 2^{b_k}.$$ 

These include, [3], [4], [6], [8], [10]. The two most advanced and recent are [10] and [3]. The complexity of [10] grows as $O(N \log(N))$ with the number of subchannels, but linearly with $B$. On the other hand [3] provides a suboptimal solution with complexity $O(N)$. Strictly speaking its complexity does not grow with $B$, as it restricts the maximum number of bits to be assigned to any subchannel to some $B^*$. Instead the complexity grows with $B^*$. The assumption of small $B^*$ is certainly problematic in subband coding, and even in communications settings when certain subchannels experience deep fades. In such a case efficiency may demand that large number bits be assigned to subchannels with more favorable conditions. A still another contributor to the complexity of [3] is the dynamic range of $\alpha_k$, which again comes into play in the presence of deep fades. All other algorithms have run times that increase with $B$.

By contrast, we provide an exact solution to (1), (2), under (3), whose complexity has an upper bound that is determined only by $N$ and is in fact $O(N \log N)$. The role of $B$ is only to induce cyclic fluctuations in the precise number of computations, and neither $B$ nor the dynamic range of $\alpha_k$, affects the upper bound of the run time.

The paper is organized as follows. Section 2 recaps a result from [13], that is specialized in this paper to formulate the algorithm given in section 3. The complexity and proof of correctness are provided in Sections 4 and 5, respectively.

2. A GENERAL RESULT

We now present a general result from [13] that solves (1), (2) for arbitrary convex $\phi_k(\cdot)$. This result is specialized to the case of (3) in subsequent sections. Denote for $k = 1, ..., N, x = 1, ..., B$,

$$\delta_k(x) = \phi_k(x) - \phi_k(x-1).$$

The $\delta_k$’s being convex, it follows that

$$\delta_k(1) < \delta_k(2) < ... < \delta_k(B), \forall k.$$ 

Let $S$ denote the set of smallest $B$ elements of

$$\tau = \{\delta_k(x) : k = 1, ..., N, x = 1, ..., B\}$$

The following lemma from [13], gives an optimum solution to (1), (2).

**Lemma 1** The optimal solution $b_k^* = [b_1^*, ..., b_N^*]^T$ to problem (1), (2), is defined as follows

$$b_k^* = \begin{cases} 0 & \delta_k(1) \notin S \\ B & \delta_k(B) \in S \\ y & \delta_k(y) \in S, \delta_k(y+1) \notin S \end{cases}$$
In essence this lemma provides a conceptual framework for solving (1), (2). Specifically, construct \( S \), and for each \( k \), determine the largest integer argument \( b_k \), for which \( \delta_k(b_k) \) is in \( S \). For general convex functions \( \phi_k \), the complexity of all known solutions grows with \( B \). In the rest of paper we present an algorithm for the convex functions of the type (3) whose complexity does not depend on \( B \).

3. PROPOSED LOADING ALGORITHM

In the case of (3), one finds that, with

\[
\beta = 1/4,
\]

\[
\delta_k(x) = \alpha_k \beta^k (\beta - 1).
\]

The first step of the algorithm requires ordering the \( \alpha_k \), and can be accomplished in \( O(N \log N) \) steps. Henceforth assume without sacrificing generality that:

\[
\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_N
\]

Define the sequence:

\[
l_i = \left\lfloor \log_\beta \left( \frac{\alpha_i}{\alpha_1} \right) \right\rfloor, \quad i = 1, 2, \ldots, N
\]

with \( l_{N+1} = \infty \), where \( \lceil a \rceil \) is the smallest integer greater than or equal to \( a \). The significance of the integers \( l_i \) is explained by Lemma 2

Lemma 2 With \( l_i \) defined in (10),

\[
\delta_1(l_i) < \delta_i(1) \leq \delta_1(l_i + 1).
\]

Proof: From (10) we have \( l_i = \left\lfloor \log_\beta \left( \frac{\alpha_i}{\alpha_1} \right) \right\rfloor \). The definition of the ceiling function gives us the following result,

\[
l_i - 1 < \log_\beta \left( \frac{\alpha_i}{\alpha_1} \right) \leq l_i.
\]

As \( \beta < 1 \) we have the following

\[
\alpha_1 \beta^{l_i-1} > \alpha_i \geq \alpha_1 \beta^{l_i}.
\]

Multiplying throughout by \( (\beta-1) \) we obtain the result (observe that \( \beta - 1 < 0 \)).

Then the proposed algorithm for solving (1), (2) under (3) is given below. It assumes that the ordering implicit in (9), has already occurred, and assigns \( b_i \) bits to the \( i \)-th subchannel.

Proposed algorithm

Step-1: Find the smallest \( k \) such that

\[
R_k = \sum_{i=1}^{k-1} (l_i - l_i) \geq B
\]

Then

\[
b_i = 0 \quad \forall i \in \{ k, k + 1, \ldots, N \}.
\]

Step-2: Find

\[
\Delta = B - R_{k-1}
\]

\[
r = \Delta \mod (k - 1)
\]

\[
q = \Delta \text{div}(k - 1)
\]

Step-3: Find the \( r \) smallest elements of the set

\[
\{ \delta_1(l_k - l_1), \delta_2(l_k - l_2), \ldots, \delta_{k-1}(0) \}.
\]

In particular, with \( l_j \) such that with \( l_j \in \{ 1, 2, \ldots, k - 1 \}, \)

\[
\delta_j, (l_k - l_j) \leq \delta_j, (l_k - l_{j+1}),
\]

call

\[
J = \{ j_1, j_2, \ldots, j_r \}.
\]

If \( r = 0 \), \( J \) is empty.

Step-4: For all \( i \in \{ 1, 2, \ldots, k - 1 \}, \)

\[
b_j = \begin{cases} l_{k-1} - l_i + q + 1 & \text{if } i \in J, \\ l_{k-1} - l_i + q & \text{else}. \end{cases}
\]

4. COMPLEXITY

Observe that the complexity implicit in achieving (9) is \( O(N \log N) \). Determination of \( k \) so that (12) holds requires at most \( 2N \) operations, regardless of \( B \). Indeed one has, with

\[
\rho_1 = 0
\]

\[
\rho_n = \rho_{n-1} + l_n,
\]

\[
R_n = (n-1)\rho_n - \rho_{n-1}.
\]

The only impact that \( B \) has in the complexity of determining \( k \) is that for sufficiently small \( B, k < N \) and the number of computations is further reduced to \( 2(k-1) \). Determining the ranking manifest in (18) is determined only by \( r \) and \( k \), and is

\[
O(r \log(k-1)) \leq O((N-1)\log(N-1)).
\]

Determination of \( r \) requires 2 operations, independent of \( B \). \( B \) does affect the precise value of \( r \), which however is no greater than \( N - 1 \).

Thus the overall complexity, is bounded by \( O(N \log(N)) \), with \( B \) playing no role in the determination of this bound. The only effect that \( B \) has on the overall complexity is to cause fluctuations in the precise number of operations, within a range that is independent of \( B \). To recap, these fluctuations occur when:

- For small \( B, k < N \), and finding \( k \) requires only \( 2(k-1) \) operations.
- As \( B \) changes \( r \) fluctuates between 0 and \( N - 1 \), and the number of operations required to determine the smallest \( r \) elements of the set in (17) changes.

5. PROOF FOR CORRECTNESS

We now show that the algorithm in section 3 does indeed solve (1), (2), under (5). In view of Lemma 1 it suffices to show that the set

\[
S^* = \{ \delta_1(1), \ldots, \delta_{k-1}(b_1), \delta_1(1), \ldots, \delta_{k-1}(b_2), \ldots, \delta_{k-1}(b_{k-1}) \}
\]

is such that

\[
S^* = S,
\]

defined in section 2. This in turn requires the demonstration of the following facts.

A) \( |S^*| = |S| = B \), where \(|.|\) represents the cardinality of its argument.
(B) For all \(i, j \in \{1, 2, \cdots, N\}\),
\[
\delta_i(b_i + 1) \geq \delta_j(b_j).
\]
The first theorem proves (A).

**Theorem 1** With \(b_i\) defined in (12-20), \(|S^*| = B\).

**Proof:** Since \(b_i = 0\) for all \(i \in \{k, k + 1, \cdots, N\}\), we need to show that
\[
\sum_{i=1}^{k-1} b_i = B.
\]
From (12-20) we have that
\[
\sum_{i=1}^{k-1} b_i = \sum_{i \in J} b_i + \sum_{i \in \{1, \cdots, k-1\} - J} b_i.
\]
\[
= r(q + 1) + (k - 1 - r)q + \sum_{i=1}^{k-1} (l_{k-1} - l_i)
\]
\[
= \Delta + R_{k-1}
\]
\[
= B.
\]
To prove (B) we need an additional Lemma.

**Lemma 3** With \(l_i, k\) and \(q\) as in (10-16),
\[
q \begin{cases} 
\leq l_k - l_{k-1} & \text{if } r = 0 \\
< l_k - l_{k-1} & \text{if } r \neq 0
\end{cases}
\]

**Proof:** From (12-16)
\[
(k-1)q + r \leq R_k - R_{k-1}
\]
\[
= \sum_{i=1}^{k} (l_k - l_i) - \sum_{i=1}^{k-1} (l_{k-1} - l_i)
\]
\[
= (k-1)(l_k - l_{k-1}).
\]
Hence the result.

We now prove (B) for the case where \(r = 0\).

**Theorem 2** Consider (10-20). Suppose \(r = 0\). Then (B) above holds.

**Proof:**
From Lemma 2 and (11) (multiplying (11) throughout by \((\beta - 1)\)) we have:
\[
\delta_i(b_i) = \alpha_i \beta^{k-1-l_i+q-1}(\beta - 1) \leq \alpha_1 \beta^{k-1+q-1}(\beta - 1) = \delta_1(b_1),
\]
This shows that \(\delta_1(b_1)\) is the largest member of \(S^*\) in (21). From Lemma 2 and (11), for all \(i \in \{1, \cdots, k\}\),
\[
\delta_i(b_i + 1) = \alpha_i \beta^{k-1-l_i+q}(\beta - 1) > \alpha_1 \beta^{k-1+q-1}(\beta - 1) = \delta_1(b_1).
\]
Following the same argument as before from (13), Lemmas 2 and 3 that for all \(i \in \{k, k + 1, \cdots, N\}\),
\[
\delta_1(b) = \alpha_1 \beta^{k-1+q-1}(\beta - 1) \leq \alpha_1 \beta^{k-1}(\beta - 1) < \alpha_k = \delta_k(1)
\]
Equations (22), (23) and (24) prove the result.

Finally we prove (B) for the case where \(r \neq 0\).

**Theorem 3** Consider (10-20). Suppose \(r \neq 0\). Then (B) above holds.

**Proof:**
With the indices \(j_i\) defined in (18), we first show that
\[
\delta_{j_i} \geq \delta_i(b_i) \quad \forall i \in \{1, \cdots, k - 1\}.
\]
In view of (18) this is clearly true for \(i \in J\). Now consider \(p \in \{\{1, \cdots, k - 1\} - J\}\).
As a result of (20), Lemma 2 and (11) (multiplying (11) throughout by \((\beta - 1)\))
\[
\delta_p(b_p) = \alpha_p \beta^{k-1-l_p+q-1}(\beta - 1)
\]
\[
\leq \alpha_{J_1} \beta^{k-1+q-1}(\beta - 1)
\]
\[
< \alpha_{J_1} \beta^{k-1-l_1+q}(\beta - 1)
\]
\[
= \delta_{j_1}(b_{j_1})
\]
\[
\leq \delta_{j_1}(b_{j_1}),
\]
where the last inequality once again follows from (18).
For all \(i \in \{1, \cdots, k - 1\} - J\), (18, 19) demonstrate that
\[
\delta_i(b_i + 1) \geq \delta_{j_1}(b_{j_1}).
\]
Further, from Lemma 2 for all \(i \in J\),
\[
\delta_i(b_i + 1) = \alpha_i \beta^{k-1-l_i+q+1}(\beta - 1) > \alpha_1 \beta^{k-1+q}(\beta - 1) \geq \delta_1(b_1).
\]
Then the result is proved by observing from Lemma 3 that
\[
\delta_{j_i}(b_{j_i}) = \alpha_i \beta^{k-1-l_i+q}(\beta - 1) \leq \alpha_{j_1} \beta^{k-1+q-1}(\beta - 1)
\]
\[
< \alpha_1 \beta^{k-1}
\]
\[
< \alpha_k = \delta_k(1).
\]

6. CONCLUSIONS
We presented an optimum bit loading algorithm with a run time of \(O(N \log N)\) which is more efficient than the ones existing in the literature, in that its complexity does not depend on the total number of bits to be allocated. The improvement in performance is very significant if \(B\) is large when compared to \(N\).

7. REFERENCES


