SPARSE AUDIO CODING VIA TARGETED DITHERING AND COMBINATORIAL DECODING

F. Mustiere[1,2], *H. Najaf-Zadeh*[1], *R. Pichevar*[1], *H. Lahdili*[1], *L. Thibault*[1], *M. Bouchard*[2]

[1]Communications Research Centre, Advanced Audio Systems
3701 Carling Avenue, K2H 8S2, Ottawa, Canada

[2]School of Information Technology and Engineering,
University of Ottawa, Ottawa, ON, K1N 6N5, Canada.

ABSTRACT

We present a novel paradigm for sparse audio signal coding. After annihilating unperceivable components in some transform domain, the encoder buffers the resulting sparse vector into small non-overlapping frames. In each frame, the active elements' amplitudes are quantized, and with the help of *a priori* known unquantized "filler" vectors (whose values are placed in inactive positions), their position is encoded such that a certain function f of the filled vector is nearly integer valued. For this purpose, the quantized values of the sparse frames are pre-adjusted in a controlled manner with distortion in mind (hence the name "targeted dithering"). The decoder then progresses through the possible combinations of the nonzero elements, and verifies with the filler vector which of these combinations produces an integer valued f, thereby retrieving the active elements' positions. In preliminary tests, good quality can be obtained by encoding 44.1 kHz signals with less than 50 kbps.

1. INTRODUCTION

This paper addresses the problem of encoding together both the position and the values of nonzero/active elements in a so-called *sparse* vector, that is, composed of a majority of zero components. Such a situation arises in various applications; most notably, many physical signals can be represented as linear combinations of only a few elementary functions, carefully selected amongst a potentially large collection of them. As an important example of generating and handling sparse vectors in the real-world, transform audio or image coding systems are currently extensively used in mainstream areas such as digital television/ telephony or data archiving/compression. As an illustration, audio coding systems are essentially based on the following ideas [1,2]. Given a vector $\mathbf{n}$ representing the signal of concern, the first step consists of formulating $\mathbf{n}$ in some transform domain, via a transformation function $\psi(.)$ as follows:

$$\mathbf{x} = \psi(\mathbf{n}) \tag{1}$$

where the resulting length-L vector $\mathbf{x}$ contains only $M \ll L$ significant elements, where the level of significance is determined by either some perceptual or mathematical criteria (Strictly speaking, L may be larger than the initial length of $\mathbf{n}$, and in fact $\mathbf{x}$ may be multi-dimensional, but we assume in the context of this paper that higher-dimensional data has been vectorized). The property of $\mathbf{x}$ only containing $M \ll L$ non-negligible elements is referred to as its *sparseness*, and is at the foundation of transform coding.

Practically speaking, having chosen ψ and possibly M in advance, the steps for encoding and storing the signal $\mathbf{n}$ are often similar to:

1. Transforming $\mathbf{n}$ into $\mathbf{x}$,
2. Perceptually/mathematically thresholding the vector $\mathbf{x}$ to retain only M non-zero values, yielding $\hat{\mathbf{x}}$.
3. Properly quantizing the M non-zero values of $\hat{\mathbf{x}}$, and then encoding them as well as their position.

At the receiver (or at playback in the context of audio), the procedure amounts to decoding the quantized values and positions, placing them into a vector of length L and then applying the inverse transformation to recover an approximation $\hat{\mathbf{n}}$ of $\mathbf{n}$. One of the unfortunate problems inherent to handling and transmitting sparse vectors, regardless of how they were obtained, lies in the potentially large overhead required to encode the positions of the active elements [3] – a uninformed approach would require a binary vector of length $\lceil \log_2 \binom{L}{M} \rceil \leq L$. In this paper, we propose to embed some of the position information within the quantized non-zero values of $\hat{\mathbf{x}}$ so as to reduce the size of the overhead. To do so, these quantized values are adjusted at the encoder so as to minimize a certain function (described below), which depends on both these values and the positions of the active elements in $\mathbf{x}$. Of course, the values cannot be blindly adjusted and we suggest a way of performing the optimization in a "perceptually controlled" manner. The method is therefore combinatoric, in the sense that the decoder must progress through the (reduced) set of possible positions, and verify which position minimizes the above mentioned function given the pre-adjusted quantized values.

The paper is organized as follows. In Section 2, the encoder side is described in a generic manner, with three steps detailed and emphasis on the pre-adjustment of the quantized values. Next, in Section 3, the decoder side is explained. Finally, Section 4 some experimental results are shown with a preliminary setup, showing the potential of the paradigm.

2. ENCODER

2.1 Sparse representation in a transform domain

The first two "traditional" transform coding steps broadly presented in our introduction are followed here as well, although we insist that the solution proposed in this paper can be applied regardless of the way that the working sparse vector is obtained. Nevertheless, using the notation introduced in the introduction, in most situations depending on the transformation chosen the goal is to assimilate $\mathbf{x}$ to a sparse vector by annihilating as many of its components as possible while retaining perceptual transparency. Representing physical signals into a "sparse domain" is an active research field; one can find examples of useful transformations in [1, 4, 5]. When choosing a transformation, there are several factors to

 ISSN 2076-1465

consider in order to reach a compromise suitable for compression. For example, while a certain representation might be sparser than an other one, it might also require a greater precision (i.e., a finer quantization) and therefore more bits per active element.

2.2 Decomposition into small non-overlapping frames

Due to the combinatorial nature of the decoder, it is recommended here to further decompose (after transformation) the obtained sparse vector $\mathbf{x}$ into non-overlapping subframes of smaller length, so as to reduce the total amount of possible positions (i.e. the search space for the decoder) to a manageable number[1]. However, there is once again an acceptable compromise to reach: first, as it will be seen below, a header is required on each subframe, and therefore multiplying the subframes may also increase the global overhead. Secondly, with overly small subframes, the likelihood of a possible transparent pre-adjustment is decreased, as several subframes will only contain few active elements to pre-adjust (the reason behind this will be clear after the next Section).

2.3 Pre-adjustment or "smart dithering"

In this Section, we propose a way to embed some information about the position of each nonzero element into their amplitude. To do this, for a given subframe $\mathbf{s}$ of length N and K active elements, let us define the following:

- Let the vector $\mathbf{p}_j$, $j \in \{1,\ldots,\binom{N}{K}\}$ contain K distinct integers in increasing order ranging from 1 to N. In other words, $\mathbf{p}_k$ is a possible support vector, also referred to as *position vector* below.
- $\mathbf{s}_q$ is the quantized version of $\mathbf{s}$. Moreover, let integer i denote the i^{th} quantization level, and $q(i)$ its corresponding real value.
- Next, let $\hat{\delta}$ be a length K vector defined relatively to $\mathbf{s}_q$ as follows. If $\mathbf{p}_j$ is the position vector associated to $\mathbf{s}_q$, then for every $k \in \{1,\ldots,K\}$, if $\mathbf{s}_q(\mathbf{p}_j(k)) = q(i)$, then $\hat{\delta}(k) = q(i+1) - q(i)$. Similarly, $\check{\delta}(k) = q(i-1) - q(i)$.
- Let $\bar{\mathbf{v}}$ be a length N, real-valued, and "full" vector (containing no zero elements) that is considered to be *a priori* known (its choice will be discussed later on). Accordingly, define $\mathbf{v}$ relatively to $\mathbf{s}_q$ and $\bar{\mathbf{v}}$ as follows. For every n, if n is contained in $\mathbf{p}_j$, then $\mathbf{v}(n) = 0$, else $\mathbf{v}(n) = \bar{\mathbf{v}}(n)$. In the following, $\mathbf{v}$ will be termed a *filler vector*.
- Let T denote a selected *target number* (the choice of which will be further discussed below), in our context an integer, and ε denote a small positive real number.

Now, define an arbitrary "targeting" function $f(\mathbf{s}_q,\mathbf{v}) = T + e$, designed such that the expected error e is relatively small to begin with. For example, our implementations use either $f(\mathbf{s}_q,\mathbf{v}) = \sum_n (\mathbf{s}_q(n) + \mathbf{v}(n))$ or $f(\mathbf{s}_q,\mathbf{v}) = \sum_n \left(|\mathbf{s}_q(n)| + |\mathbf{v}(n)|\right)$, and $T = [f(\mathbf{s}_q,\mathbf{v})] \in \mathbb{N}$ (the value of the nearest integer to $f(\mathbf{s}_q,\mathbf{v})$). The goal is here to minimize e by adjusting in a controlled manner the values of $\mathbf{s}_q$. Before explaining how to do this, let us show why this is precisely why the decoder will be able to retrieve $\mathbf{p}_j$. Suppose that $\mathbf{s}_q$ has been successfully adjusted, and accordingly e is very small. The K values of the ajusted $\mathbf{s}_q$ are then sent to the decoder. Then, with *only* these K values, the decoder can go through the set of possible positions, from each of which a particular $\mathbf{v}$ is defined, and stop whenever it meets one such that $f(\mathbf{s}_q,\mathbf{v}) - [f(\mathbf{s}_q,\mathbf{v})]$ is smaller than a certain threshold.

In order to minimize e, we propose to cast the problem as a binary linear program as follows. Let Φ and $\mathbf{y}$ be defined as:

$$\Phi = \begin{bmatrix} \mathbf{I}_K & \mathbf{I}_K \\ \hat{\delta} & \check{\delta} \\ -\hat{\delta} & -\check{\delta} \end{bmatrix} \text{ where } \mathbf{I}_K \text{ is the } K \times K \text{ identity matrix} \quad (2)$$

$$\mathbf{y} = [1,1,\ \ldots\ 1, T - f(\mathbf{s}_q,\mathbf{v}) + \varepsilon, -T + f(\mathbf{s}_q,\mathbf{v}) + \varepsilon]^T \quad (3)$$

Then, the idea consists in solving for the $2K$-long binary vector $\hat{\mathbf{u}}$ as the solution of the problem below:

$$\arg\min \|\mathbf{u}\|_0 \text{ subject to } \Phi\mathbf{u} \leq \mathbf{y} \quad (4)$$

Then, the vector $\mathbf{s}_q$ can then simply be adjusted by increasing by one level its values corresponding to the nonzero entries in the first K elements of $\hat{\mathbf{u}}$, and decreasing by one level those corresponding to the nonzero entries in the last K elements of $\hat{\mathbf{u}}$.

Several remarks must be made regarding the problem given in Eqn. 4:

- While the problem is *NP*-complete, it can be solved efficiently by Implicit Enumeration, or the Balas Additive Algorithm, originating from [6]. The Balas Additive Algorithm is in essence a branch-and-bound method based on efficient heuristics.
- From its formulation, the problem is always feasible for $\varepsilon = e$ (the solution is then the trivial solution $\mathbf{u} = 0$). Ideally, ε should however be as low as possible – but if a certain choice for ε makes the problem infeasible, another slightly larger value can be attempted, and so on until the problem is solvable.
- As shown in Eqn. 4, we seek the solution that minimizes the required changes in $\mathbf{s}_q$ (with binary vectors, note that the 0-norm is identical to the 1-norm). In particular, this excludes other solutions obtained from needlessly exchanging two initially adjacent levels.
- Moreover, it is simple to control which element can be adjusted or not: excluding a certain element simply means that the vector $\mathbf{u}$ to solve for has a reduced dimension (by 2 elements). In the same vein, a certain element can be allowed to be adjusted only towards one direction.
- From the form of Eqns. 2 and 3, as required, one single element of $\mathbf{s}_q$ cannot be simulaneously increased and decreased (in other words, the first and last K elements of $\mathbf{u}$ are mutually exclusive).
- If the transformation of Eqn. 1 allows for a larger error tolerance than one quantization step in the amplitude of some or all components, the procedure can be repeated, each time excluding the elements whose amplitude have been maximally modified. The initial value $f(\mathbf{s}_q,\mathbf{v})$ is simply recomputed at the beginning of each new iteration.

In Figure 1, a certain vector of length 24 is subjected to the above procedure, and the optimization is successful in driving the sum of quantized values very close to the integer 2. It is clear that if (*i*) the quantization is very coarse, and/or (*ii*) the vector $\mathbf{s}$ is very sparse, then one can think of many examples where the above procedure will not succeed satisfactorily. To address the first concern, we propose to include a

[1] whether a number is deemed "manageable" depends of course on the context and the available computational resources.

header with some elements of information regarding the particular error that should be observed by the decoder when the correct position vector is being tested (see Section 2.5). Alternatively, a certain bank of appropriate filler vectors, *a priori* known to both the encoder and the decoder, may be used to select one that will require the least amount of changes and guarantee that the procedure will succeed (some discussion regarding filler vectors is given in Section 2.4). Regarding the second concern, as discussed in Section 4, this procedure is chosen to be only implemented for cases with K roughly between $0.3N$ and $0.7N$, where the "direct" encoding of the active positions would begin to be the costliest. In the context of small subframes (as prescribed in Section 2.2), when there are too little active elements (respectively more than a certain amount), it is advantageous to encode directly the positions of the non-zero elements (respectively of the zero elements).

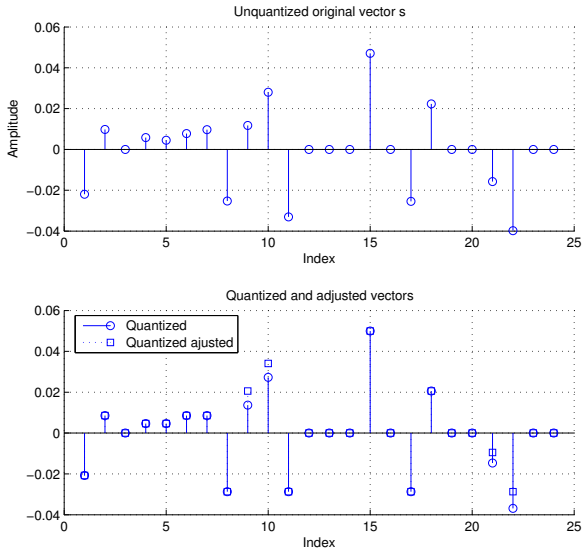

Figure 1: Example of pre-adjustment of a quantized vector. On the top graph, the original, unquantized vector is shown. On the bottom graph, both the quantized and the quantized "adjusted" vectors are shown. In this particular case, 8-bit scalar quantization is used, and only one round of optimization was applied with $\varepsilon = 5 \times 10^{-5}$ and $T = 2$. The filler vector used $\mathbf{v}$ is normally distributed around 0, and the unadjusted vector has $f(\mathbf{s}_q, \mathbf{v}) \simeq 2.28987$. For the adjusted signal, this value is about 1.99997.

2.4 Choice of filler vectors and targeting function

For the given "smart dithering" procedure to be able to find an appropriate solution with a single or few iterations, the choice of a targeting function and the design of appropriate filler vectors is important. An "appropriate" filler vector $\mathbf{v}$ is defined, relatively to the targeting function f and the amplitudes in $\mathbf{s}_q$, as one which will not only render the optimization problem in Eqn. 4 feasible with the smallest ε and the smallest amount of changes in $\mathbf{s}_q$, but also will yield significantly different function values for an incorrect positioning of the active elements in a blank length-N vector. While we have reached an operational solution, this problem is currently under investigation. We recapitulate here a few findings and ideas:

- Given f, these appropriate filler vectors must depend on the statistics of $\mathbf{s}_q$. While our current implementation uses a fixed set of filler vectors (up to 8), the idea of adaptive filler vectors is therefore appealing.
- At this point, we heuristically find that if $\mathbf{s}_q$ is approximately zero-mean and has a variance of σ^2, then drawing $\mathbf{v} \sim \mathcal{N}(0, 10 \times \sigma^2)$ can provide some good robustness. A relatively larger variance ensures that incorrect positions will be unequivocally excluded by the decoder.
- To guarantee that an appropriate filler vector will be available during encoding, a small bank of 8 vectors drawn from $\mathcal{N}(0, 10 \times \sigma^2)$ are used in our implementation. Thus, currently three bits are required to be included in a header for each subframe. The design and unsupervised adaptation of a single filler vector would eliminate the need for these three bits.

2.5 Including information in headers

Supposing the optimization above was sufficiently successful to guarantee the recovery of the positions (Figure 1 represents an example of such a case), then the decoder must simply know how many active elements are present in the incoming subframe. Noting that the encoder can easily verify whether the decoder will be able to correctly retrieve the positions or not, more information can be included if necessary. For example, some hints on the number of trials required to reach the correct solution can be provided. If the quantization is relatively coarse, the optimized value of $f(\mathbf{s}_q, \mathbf{v}) - [f(\mathbf{s}_q, \mathbf{v})]$ may either still be relatively large and/or another small enough error might be encountered – for an incorrect position vector – before the correct one is tested. In such cases, it is proposed to provide in a header some information about the particular error corresponding to the right position vector. For example, in one of our implementations, we encode the exponent of the error seen during encoding. Additionally, depending on the case it may help to let the decoder know what the target integer is. Finally, as explained in the previous Section, a bank of 8 seeds is used during encoding, and thus the decoder must know which one was used for each subframe. In Section 4.2, some ideas to encode efficiently such information in an operational solution are given.

3. DECODER

The decoder is in essence very simple, and is described as follows. For each subframe, it must:

1. read the header to determine the amount K of active elements, as well as any other information if present (see Section 2.5),
2. decode the K amplitudes of the nonzero elements,
3. go through the list of possible position vectors $\mathbf{p}$: for each possibility, it must extract the corresponding filler vector $\mathbf{v}$, place the K amplitudes in a vector $\mathbf{s}$ and test how close $f(\mathbf{s}, \mathbf{v}) - [f(\mathbf{s}, \mathbf{v})]$ is to zero – until a low-enough error is encountered.

A certain type of list exhaustion must be defined and accordingly, both for complexity evaluation and for potential header information embedding, the required amount of trials to reach the correct solution must be determined. For simplicity, to go through the possibilities, the procedure is

Trial number	Configuration	Trial number	Configuration
1	11000	6	01010
2	10100	7	01001
3	10010	8	00110
4	10001	9	00101
5	01100	10	00011

Table 1: An example of a possible list exhaustion order for the decoder

defined in Table 1 on an example with $K = 2$ and $N = 5$. According to the above-defined procedure for progressing through the possibilities for the position vector, given a certain configuration the corresponding trial number can be obtained back as follows. For a given $\mathbf{p}$, let $\mathbf{z}$ be a length K vector containing the number of zeros at the left of each nonzero element (for example, for the configuration 010010, $\mathbf{z} = [1,2]$). Define also $c_i = \sum_{k=1}^{i-1} \mathbf{z}(k)$ (with $c_1 = 0$). Then, the number of trials required is:

$$n_t(\mathbf{p}) = 1 + \sum_{i=1}^{K} \sum_{j=1}^{\mathbf{z}(i)} \binom{N-i-j-c_i+1}{K-i} \qquad (5)$$

From Eqn. 5, it is clear that the first elements of $\mathbf{z}$ have a very large impact on $n_t(\mathbf{p})$, whereas the last few have much less impact on it. In practice, it can be advantageous for the encoder to reserve a bit for flipping or circularly shifting the current subframe, so as to reduce the amount of trials wasted by the decoder.

4. SOME RESULTS WITH A PRACTICAL SETUP

It is proposed to encode a 7.6-seconds speech signal with a sampling frequency of 44.1 kHz. In this test signal, a female speaker utters the following sentence: *"To administer medicine to animals is frequently a very difficult matter, yet sometimes it's necessary to do so"*.

4.1 Perceptual thresholding in the MDCT domain

In order to obtain a sparse representation for the input signal, it is first decomposed by means of a Modified Discrete Cosine Transform (MDCT) using frames of length 1024 and a Kaiser-Bessel derived window. Next, the simultaneous frequency masking threshold of the signal is obtained. More specifically, the threshold used in the ISO MPEG-1 layer 1 psychoacoustic model 1 is used (see [2] for a description of both the MDCT and the steps involved in the computation of the masking threshold). Using the relationship between the DFT and the MDCT (see [7]), all the MDCT coefficients below the masking threshold are set to 0. With this method, a nearly-transparent audio quality can be achieved by dropping up to 75% of the coefficients. In our specific speech-only case, only 30406 coefficients are retained (for an initial length of 335348). During the above perceptual thresholding, each remaining coefficient is also tagged with an importance factor determined by the distance from the masking threshold. Then, the encoder is instructed not to adjust the quantized values of the most perceptually important MDCT coefficients.

4.2 A possible bit allocation scheme

In our preliminary tests, the entire MDCT matrix obtained in the previous Section is vectorized and decomposed into small subframes of length N. Specifically, the subframes have a length $N = 24$. Such a small number ensures that in the worst case scenario, about 2.7 million iterations will be required by the decoder. However, with the dedicated bit allocation described below, the average amount of iterations that is observed in our tests is about 150000.

The header begins with the following information (referred to as the "a" bits below):

- "00" to indicate that there are no active elements in the current subframe.
- "01" to indicate that there are no active elements in the next 6 subframes.
- "1" to indicate that there is at least one active element in the current subframe.

The above is advantageous for the MDCT transformation described in 4.1, as there are often multiple consecutive inactive subframes.

In the case where at least 1 element is present, the header is followed by 5 bits (b_1 to b_5). If b is the integer value of $b_1b_2b_3b_4b_5$, then $b = i-1$, $1 \leq i \leq 24$ indicates that there are i nonzero elements in the subframe. Additionally:

- $i = 25$ indicates that there are 12 active elements *and* the first element is in the first position.
- Similarly, with $i = 26$ up to 28 indicates that the first element is in the second up to fourth position. This implies that $i = 12$ means that the 12 active elements are located *after* the fourth position.
- Moreover, $i = 29$ resp. 30 indicate that there are 11 active elements and the first element is in the first resp. second position. Again, $i = 11$ then means that the 12 active elements are necessarily located after the second position.
- Finally, $i = 31$ and 32 give the same information for 13 elements.

The above considerably reduces the number of iterations required at the decoder. Referring to Eqn. 5, the above reduces the value of $\mathbf{z}(1)$ for the three most problematic cases (i.e., the three cases with the largest search space) of 11, 12, and 13 active elements within the subframes of length 24.

Next, a single bit c indicates whether or not the subframe was flipped by the encoder (whichever resulted in a smaller required amount of iterations for the decoder). Moving along, a set of "d" bits then contains some information about the position of the nonzero elements. Here, because of the fact that directly encoding the position of the nonzero (resp. zero) elements is fairly inexpensive for $1 \leq K \leq 4$ (resp. $20 \leq K \leq 23$), it is chosen to only apply the "smart dithering" method for $5 \leq K \leq 19$. For $2 \leq K \leq 4$ (resp. $20 \leq K \leq 22$), $3K$ (resp. $3(N-K)$) bits are assigned to pointing to the locations of the nonzero (resp. zero) elements by Run-Length-Encoding (RLE). When $K = 1$ or $K = 23$, 4 bits are used. Bit c is still consistent in that it indicates from which end of the subframe to begin the RLE. Clearly, if $K = 24$ no bits "d" are required at all. When $5 \leq K \leq 19$, the "d" bits represent some partial information about the position vector. As previously mentioned in Section 2.5, it is chosen here to use a bank of 8 filler vectors, and to

facilitate decoding we also send the absolute value of the target integer. Three bits are used to point to the right filler vector, and two bits are used for the absolute value of the target integer (from 0 to 3 or more).

Subsequently, the K required amplitudes are concatenated and the subframe code ends. Regarding the amount of bits required for representing these amplitudes, at this point we propose the following: for $5 \leq K \leq 10$, 8 bits are used, and otherwise 7 bits are used. The reason behind this choice is related to the observation that the sum of (filled) sparser vectors is more difficult to drive to integer values, due to the reduced amount of possible changes. This is counterbalanced in our implementation by the use of more quantization levels.

4.3 Simulation results

For the speech signal under consideration, after perceptual thresholding and encoding, the distribution of frames and bit count is given in Table 2. According to Table 2, the total bit

K	Subframes	Req. bits	K	Subframes	Req. bits
0	9908	6546	13	158	16274
1	412	7416	14	149	16390
2	352	9504	15	115	13455
3	309	11433	16	96	11904
4	311	14617	17	69	9039
5	289	15028	18	34	4692
6	264	15840	19	27	3915
7	295	20060	20	13	2067
8	223	16948	21	8	1304
9	276	23184	22	4	668
10	249	22908	23	5	90
11	211	18779	24	0	0
12	218	20928	**Total**	13995	282989

Table 2: Distribution of active elements. When $K = 0$, the encoder finds a total of 1327 blocks of 6 consecutive empty frames .

count is 282989, which amounts to approximately 37 kbps.

As the next Section will outline, there is significant room for improvement in multiple areas, nevertheless our informal listening tests indicate that the recovered speech with adjusted subframes is nearly undistinguishable from a reference case obtained using unadjusted subframes.

5. CONCLUSION AND FUTURE WORK

We proposed in this paper a different approach to audio signal transform encoding/decoding. A certain position-dependent structure is forced upon the quantized amplitudes of the nonzero elements within small, non-overlapping frames, thereby creating a detection criterion for the decoder that is embedded in these values. To do this, a so-called targeting function, accompanied by a filler vector, are designed and used to choose a target value for the quantized amplitudes. The problem of adjusting these amplitudes in a perceptually reasonable manner is posed in terms of a Binary Integer Linear Program, which can be efficiently solved while applying constraints on the tolerable error in each amplitude to adjust. The decoder can then test possible combinations of positions and picks the correct one by computing for each possibility the value of the targeting function and verifying that it is close to the expected target value. There are many areas of improvements currently being investigated:

- A study of the appropriateness of the targeting function f and filler vectors $\mathbf{v}$ is required; doing so will accelerate the optimization, minimize the required adjustments and shorten the subframe headers. In parallel, it may not be the best choice to require integer targets; rather, the encoder could encode most common target values.
- The decoding method is, at this stage, computationally demanding. One could however think of more adequate ways of exploring the number of possibilities. For example, given the read amplitudes of the nonzero components and the knowledge of the filler vectors, several possibilities could be directly excluded. This could be further reduced if the target value is known by the decoder.
- Rather than including additional information in headers in order to guarantee that the decoder will successfully retrieve the correct positions (see Section 2.5), it may be more advantageous to merely include one bit to indicate whether the operation will or will not succeed without extra information. In the latter situation, the encoder could then resort to traditional techniques such as RLE. The above depends on an analysis of the proportion of problematic subframes.
- In order to further reduce the header size, it might be advantageous to exploit the structure of the sparse vectors – for example, in the MDCT matrix case, neighbour columns are often correlated.
- Finally, some more suitable and less primitive bit allocation schemes are being studies as well.

REFERENCES

[1] S. Mallat, *A Wavelet Tour of Signal Processing*. New York: Academic, 1999.

[2] A. Spanias, *Audio Signal Processing and Coding*, Wiley-Interscience, 2007.

[3] R. G. Baraniuk, "Compressive Sensing," Lecture Notes in *IEEE Signal Processing Magazine*, Vol. 24, No. 4, pp. 118–120, July 2007.

[4] R. Pichevar, H. Najaf-Zadeh, L. Thibault, "A Biologically-Inspired Low-bit-rate Universal Audio Coder", in *Proceedings of the AES*, Vienna, May 2007.

[5] J.P. Princen and A.B. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 34, No. 5, pp. 1153–1161, 1986.

[6] E. Balas, "An additive algorithm for solving Linear Programs with zero-one variables," in *Operatons Research*, Vol. 13, No. 4, pp. 517–546, 1965.

[7] H. Najaf-Zadeh and P. Kabal, "Improving Perceptual Coding of Narrowband Audio Signals at Low Rates", in *Proc. ICASSP 1999*, Phoenix, Arizona, March 1999, pp. 913–916.