Split Table Extension: A Low Complexity LVQ Extension Scheme in Low Bitrate Audio Coding

Jin Wang, Peilin Liu, Ji Kong, and Rendong Ying

***Abstract*—Embedded Algebraic Vector Quantization (EAVQ) is a fast and efficient Lattice Vector Quantization (LVQ) scheme used in low-bitrate audio coding. However, a defect of EAVQ is the overload distortion which causes unpleasant noises in audio coding. To solve this problem, specific base codebook extension schemes should be carefully considered. In this letter, we present a novel EAVQ codebook extension scheme—Split Table Extension (STE), which splits a vector into two smaller vectors: one in the base codebook and the other in the split table. The base codebook and the split table are designed according to the appearance probability of quantized vectors in audio segments. Experiments on encoding multiple audio and speech sequences show that, compared with the existed Voronoi Extension scheme, STE greatly reduces computation complexity and storage requirement while achieving similar coding quality.**

***Index Terms*—Embedded algebraic vector quantization, lattice vector quantization, split table extension, Voronoi extension.**

I. Introduction

LATTIC vector quantization (LVQ) has been widely used in speech, audio, image, and video coding [1], [2] due to its highly structured and efficient codebook [3]. Embedded Algebraic Vector Quantization (EAVQ) is a fast and efficient LVQ scheme that employs algebraic algorithm in the codebook design [4]. However, One defect of EAVQ is the *overload distortion* [5] problem that the large vectors which are not included by the base codebook will be quantized to the nearest vector inside the base codebook with large quantization errors. This process will introduce unpleasant noises when used in audio coding. To solve the problem, the Voronoi extension (VE) scheme is proposed in [6]. It represents a large vector outside the base codebook with the sum of a scaled base vector and a Voronoi vector. With this scheme, VE is able to indirectly enlarge the base codebook to include the large vectors. VE has been applied to the TCX codec in AMR-WB+ standard [7]. Despite its good performance in audio coding, VE has two main disadvantages. One is the complexity due to the matrix operations it employs in the computation procedure. The other is the inflexible bit allocation for each of the Voronoi vector components, which may incur low coding efficiency.

Manuscript received May 31, 2009; revised September 04, 2009. First published September 22, 2009; current version published October 21, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Saeid Sanei.

The authors are with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: wangjin.jane@gmail.com; liupeilin@sjtu.edu.cn; johnhophen@sjtu.edu.cn; uingrd@lycos.com).

Digital Object Identifier 10.1109/LSP.2009.2032982

In this letter, we present a novel Split Table Extension (STE) scheme. The scheme only employs simple arithmetic operations such as addition and subtraction in the extension procedure and uses smaller base codebook compared with VE. It also uses flexible bit allocation in encoding the extension vector components. In STE, the overloaded vector is split into two vectors. One is in the base codebook and the other is an extension vector that is encoded by a multilevel extension table called the split table. Experimental results show that, compared with VE, the STE scheme greatly reduces computational complexity and storage requirement while having similar coding efficiency and quality.

II. Review of EAVQ and Voronoi Extension

A. RE_8-Based EAVQ

The EAVQ introduced by [4] is a highly-structured LVQ scheme which is based on the well-known rotated Gosset lattice RE_8 defined as

$$RE_8 = 2D_8 \cup \{2D_8 + (1, 1, \ldots, 1)\} \tag{1}$$

where $D_8 = \{(x_1, x_2, \ldots, x_8) : x_i \in \mathbb{Z}, (\sum_{i=1}^{8} x_i)\ mod\ 2 = 0\}$. The important properties of RE_8 lattice are summarized as follows:

1) All lattice points in RE_8 are 8-dimensional vectors.
2) All 8 components of an RE_8 vector have identical parity.
3) The component sum of an RE_8 vector $\mathbf{y} = \{y_1, y_2, \ldots, y_8\}$ is multiple of 4, i.e., $(\sum_{i=1}^{8} y_i)\ mod\ 4 = 0$.

Any RE_8 vector $\mathbf{y}$ can be generated by

$$\mathbf{y} = [k_1\ k_2 \cdots k_8] G_{RE_8} \tag{2}$$

where $k_i \in \mathbb{Z}\ (i = 1, 2 \cdots 8)$ is the index vector of the lattice point $\mathbf{y}$ and G_{RE_8} is the 8×8 generator matrix of RE_8 lattice.

EAVQ spherically shapes the RE_8 lattice and selects the innermost 11 spheres to construct the codebook consisting of six subsets $Q_0 - Q_5$. It quantizes any 8-D vector into the nearest lattice point in the codebook. This will cause the overload distortion problem when a very large vector outside the selected spherical bound is arbitrarily quantized into a small lattice point inside the codebook, which introduces intolerable noises when EAVQ is used in audio coding.

B. Voronoi Extension

VE in [6] is designed to resolve the overload distortion problem in EAVQ. In VE, the codebooks are composed of the base codebook and the extension codebook. The base codebook includes the lattice points in RE_8 with relatively

high appearance frequency, while the extension codebook is generated by extending the base codebook using VE scheme to include all other RE_8 lattice vectors. We denote the base codebook as C, base vector as $\mathbf{c} \in C$, extension order as r and r^{th} order extension codebooks as $C^{(r)}$. The VE scheme can be described as

$$C^{(r)} = 2^r C + V_{2^r} = \bigcup_{\mathbf{c}\in C, \mathbf{v}\in V} 2^r\mathbf{c} + \mathbf{v} \tag{3}$$

where V_{2^r} is the r^{th} order Voronoi space and $\mathbf{v} \in V$ is the Voronoi vector.

For a given order r, [9] defines Voronoi index vector by

$$\mathbf{k}_v = (\mathbf{y}G_{RE_8}^{-1}) \bmod 2^r. \tag{4}$$

When applied to AMR-WB+ in [7], VE uses iterations to find r and $\mathbf{v}$. It first computes $\mathbf{k}_v$ from (4) and then determines $\mathbf{v}$ from $\mathbf{k}_v$ using an algorithm described in [8]. Knowing $\mathbf{v}$, $\mathbf{c}$ can be calculated by

$$\mathbf{c} = \frac{1}{2^r}(\mathbf{y} - \mathbf{v}). \tag{5}$$

If $\mathbf{c}$ is in the base codebook, the Voronoi vector and the extension order are successfully found. Otherwise the extension order r should be increased to recalculate the Voronoi vector and the base vector. This searching process employs matrix multiplication operations with high-computational complexity in each iteration.

Voronoi vector $\mathbf{v}$ is encoded by its index vector $\mathbf{k}_v$ determined in (4). [6] proved that, given extension order r, the bits required to encode $\mathbf{k}_v$ are $r/component$. This feature of VE indicates the identical bit allocation for each vector component. However, extension usually occurs due to the large value(s) of only one or two vector component(s). For example, extension should be applied to $\{25, 1, 1, 1, 1, 1, 1, 1\}$ due to the large component value "25". In this case, the bits allocated for extending small vector components "1"s are wasteful.

III. Split Table Extension

A. Basic Principles

In the previous section, it is noted that VE employs matrix multiplication operations in the searching of r and $\mathbf{v}$, which requires high computational effort. We reduce the complexity by proposing the STE method that a large RE_8 lattice vector $\mathbf{y}$ is represented by the sum of two smaller vector $\mathbf{c}$ and $\mathbf{s}$, denoted as:

$$\mathbf{y} = \mathbf{c} + \mathbf{s} \tag{6}$$

where the vector $\mathbf{s}$ is defined as the split vector and $\mathbf{c}$ is the remainder with property $|\mathbf{c}| \leqslant |\mathbf{y}|$. The aim of STE is to find some $\mathbf{s}$ to make $\mathbf{c}$ be an RE_8 vector included in the base codebook. Observing that $\mathbf{c} = \mathbf{y} - \mathbf{s}$, to make all eight components of $\mathbf{c}$ have identical parity and $(\sum_{i=1}^{8} c_i) \bmod 4 = 0$, $\mathbf{s} = \{s_1, s_2, \ldots, s_8\}$ should also be an RE_8 vector.

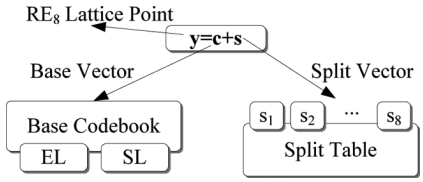

Fig. 1. Encoding RE_8 Vector with STE.

Moreover, to achieve more flexibility in bit allocation for each vector component, each component of vector $\mathbf{s}$ can be changed independently. $(\sum_{i=1}^{8} c_i) \bmod 4 = 0$ always holds no matter which components of $\mathbf{s}$ changes. Therefore, we make $s_i \bmod 4 = 0,\ i = 1, 2, \ldots, 8$.

To make coding more efficient, the sign information of $\mathbf{y}$ is stored in $\mathbf{c}$. That is, the components of $\mathbf{c}$ have the same polarity (sign) with those of $\mathbf{y}$, and $|c_i| \leqslant |y_i|$. Denoted as:

$$\begin{cases} y_i = c_i + |s_i| & y_i > 0,\ y_i \bmod 2 = 0,\ c_i > 0 \\ y_i = c_i - |s_i| & y_i < 0,\ y_i \bmod 2 = 0,\ c_i < 0 \\ y_i = c_i + |s_i| + 1 & y_i > 0,\ y_i \bmod 2 = 1,\ c_i > 0 \\ y_i = c_i - |s_i| + 1 & y_i < 0,\ y_i \bmod 2 = 1,\ c_i < 0 \end{cases} \tag{7}$$

where $i = 1, 2, \ldots, 8$. For example, if $\mathbf{y} = \{-31, 1, 1, 1, 1, 1, 1, 1\}$ and $\mathbf{s} = \{-28, 0, 0, 0, 0, 0, 0, 0\}$, $\mathbf{c}$ will be $\{-3, 1, 1, 1, 1, 1, 1, 1\}$ with the same parity and polarity of $\mathbf{y}$. To decode $\mathbf{y}$, only $\mathbf{c}$ and the absolute value of $\mathbf{s}$ are required since the polarity of $\mathbf{s}$ and $\mathbf{y}$ can be determined by that of $\mathbf{c}$.

In (7), the vector $\{1, 1, 1, 1, 1, 1, 1, 1\}$ is subtracted from an even vector $\mathbf{y}$ before applying STE. This procedure is used to avoid sign information losing for even vectors. For example, let $\mathbf{y} = \{20, 20, 20, 0, 0, 0, 0, 0\}$, if it is split into $\mathbf{s} = \{20, 20, 20, 0, 0, 0, 0, 0\}$ and $\mathbf{c} = \{0, 0, 0, 0, 0, 0, 0, 0\}$, the sign information of $\mathbf{y}$ is missing. Therefore, it should be first converted to an odd vector $\{19, 19, 19, -1, -1, -1, -1, -1\}$ and then split to $\mathbf{s} = \{16, 16, 16, 0, 0, 0, 0, 0\}$, $\mathbf{c} = \{3, 3, 3, -1, -1, -1, -1, -1\}$ to keep the sign information.

B. Base Codebook and Split Table Design

The two vectors $\mathbf{c}$ and $\mathbf{s}$ in STE are encoded by STE base codebook and the split table respectively as shown in Fig. 1.

In EAVQ, the spherical shaping of RE_8 lattice and the codebook selection are based on the assumption that all vectors are i.i.d zero-mean Gaussian [4]. However, the vectors in real source often behave non-Gaussian. Therefore, we further investigate the appearance probability of the codebook vectors in 85 MPEG2-standardized audio and speech sequences to construct the base codebook for STE.

The two *Prob* columns of Table I list the appearance probability of RE_8 vectors in test sequences, including all the vectors in the innermost six RE_8 spheres and some vectors in other spheres. The origin $\{0, 0, 0, 0, 0, 0, 0, 0\}$ has ultra-high appearance probability which dominates all other vectors, therefore it is eliminated from the probability statistics. The vectors are listed and stored in the form of *leader*, which is defined as a positive vector that can be used to generate other vectors from appropriate permutations or sign alternation of the vector components. The number of vectors that can be generated from

TABLE I
RE_8 LATTICE VECTOR APPEARANCE PROBABILITY AND BASE CODEBOOK USED IN STE

Leader	Prob. %	Category Subset	Leader	Prob. %	Category Subset
0,0,0,0,0,0,0,0		SL, Q_0	5,3,1,1,1,1,1,1	1.74	SL, Q_3
2,2,0,0,0,0,0,0	21.48	SL, Q_1	3,3,3,3,1,1,1,1	1.19	EL, Q_3
1,1,1,1,1,1,1,1	21.49	EL, Q_1	6,2,2,2,0,0,0,0	0.92	X, X
4,0,0,0,0,0,0,0	0.62	SL, Q_1	4,4,4,0,0,0,0,0	0.04	X, X
2,2,2,2,0,0,0,0	9.63	SL, Q_2	4,4,2,2,2,2,0,0	0.00	X, X
3,1,1,1,1,1,1,1	9.09	EL, Q_2	5,3,3,1,1,1,1,1	1.24	SL, Q_3
4,2,2,0,0,0,0,0	4.92	SL, Q_2	3,3,3,3,3,1,1,1	0.18	EL, X
2,2,2,2,2,2,0,0	2.23	X†, X	7,1,1,1,1,1,1,1	0.22	SL, Q_3
3,3,1,1,1,1,1,1	6.37	EL, Q_3	5,5,1,1,1,1,1,1	0.27	SL, Q_3
4,4,0,0,0,0,0,0	0.42	SL, Q_2	3,3,3,3,3,3,1,1	0.05	EL, X
4,2,2,2,2,0,0,0	3.38	X, X	8,0,0,0,0,0,0,0	0.07	SL, Q_2
2,2,2,2,2,2,2,2	0.22	X, X	3,3,3,3,3,3,3,1	0.53	EL, Q_3
5,1,1,1,1,1,1,1	1.11	SL, Q_3	3,3,3,3,3,3,3,3	0.35	EL, Q_3
3,3,3,1,1,1,1,1	2.13	EL, Q_3	9,1,1,1,1,1,1,1	0.13	SL, Q_3
6,2,0,0,0,0,0,0	0.83	SL, Q_2	11,1,1,1,1,1,1,1	0.56	SL, Q_3
4,4,2,2,0,0,0,0	1.41	X, X	13,1,1,1,1,1,1,1	0.35	SL, Q_3
4,2,2,2,2,2,2,0	0.73	X, X			

†X means the leader does not belong to any leader category or any codebook subset.

the leader is the size of the leader. For example, the vectors $\{3,1,1,-1,1,1,1,1\}$ and $\{1,1,-1,1,-3,1,-1,1\}$ have the same leader $\{3,1,1,1,1,1,1,1\}$ whose size is 1024.

From the statistics in Table I, the final base codebook is constructed. The 23 leaders in the base codebook are composed of 2 categories: the *Essential Leaders (EL)* and the *Subsidiary Leaders (SL)*. They are further divided into 4 subsets Q_0, Q_1, Q_2 and Q_3 with size 1, 256, 4096, and 65536 respectively. The two *Category/Subset* columns in Table I indicate the leader category as well as the codebook subset for each leader.

ELs include all the leaders with component value 1 or 3. From (7), c_i can be odd and less than 4 by choosing appropriate s_i, denoted as

$$c_i = y_i - s_i = \begin{cases} 1, & \text{if } y_i \bmod 4 = 1 \\ 3, & \text{if } y_i \bmod 4 = 3. \end{cases} \tag{8}$$

This feature makes the selection of ELs necessary since they guarantee that base vector can be found for every lattice vector after applying STE. ELs with high appearance probability are stored directly in the base codebook while ELs $\{3,3,3,3,3,1,1,1\}$ and $\{3,3,3,3,3,3,1,1\}$ which have relatively low appearance probability are stored in the form of swapping the digit "3" and "1" of ELs $\{3,3,3,3,1,1,1,1\}$ and $\{3,3,1,1,1,1,1,1\}$.

The remaining leaders in the base codebook are SLs which are selected based on the following three principles.

1) The SLs should be the leaders with relatively high appearance probability in Table I.
2) The total size of the leaders selected for each codebook subset should not exceed the size of that codebook subset.
3) Some even leaders with high appearance probability can be eliminated from the base codebook because they can be converted to an odd base vector by subtracting $\{1,1,1,1,1,1,1,1\}$, such as $\{2,2,2,2,2,2,0,0\}$.

The absolute split vector $|\mathbf{s}|$ is encoded by the split table. Previous discussion about VE mentions that the identical bit allocation for each component of Voronoi vector incurs inflexibility. Therefore, we propose that the 8 components of the split vector

TABLE II
SPLIT TABLE

index m	**split value** $\lvert s_i \rvert$					
	$r=0$	$r=1$	$r=2$	$r=3$	$r=4$	$r=\cdots$
0	0	4	8	16	32	
1	N/A	N/A	12	20	36	
2			N/A	24	40	
3				28	44	
4				N/A	48	
5					52	
6					56	
7					60	

$\mathbf{s}$ are independent encoded by the split table with different bit allocation.

Since $s_i \bmod 4 = 0$, the split table should include all the non-negative split values which are multiple of 4. Furthermore, since probability decreases with the increasing order of the split values, it is reasonable to design a multilevel split table in which the small split values require less bits and large split values require more. Table II shows the split table used in STE. The absolute split vector component $|s_i|$ is represented by the encoding of its order r and index m. r is encoded in the unary code, i.e., the codeword contains r bits "1" ahead of one bit "0". The index is encoded directly by $r-1$ bits (For $r=0$ and $r=1$, no bits are used to encode index). Consequently, total bits used to encode a split value is $2r$ (For $r=0$, only 1 bit is needed). Note that the order and the index can be different for each split vector component according to its value. This feature provides an efficient encoder especially for the vectors containing only a few large components. The small vector components do not need extension, therefore the split vector components for them can be encoded with only 1 bit "0" each.

A further investigation of the split table shows that the split values for a given order $r \geqslant 1$ are in the interval $[2^{r+1}, 2^{r+2})$. Therefore, we can keep track of the r and m when searching the split values and it is unnecessary to keep the split table in the memory.

The encoding of the split vector can be illustrated by an example that $\mathbf{y} = \{20,4,4,4,4,4,4,4\}$. We calculate that $|\mathbf{s}| = \{16,0,0,0,0,0,0,0\}$ and $\mathbf{c} = \{3,3,3,3,3,3,3,3\}$. The first component value of split vector is 16 which requires 6 bits to encode ($r = 3$). The remaining components are all 0 requiring 1 bit each. Consequently, the total bits required to encode the split vector are 1 (the parity information) $+6+1\times 7 = 14$. But for VE, it can be calculated that $r=2$ and $2\times 8+2(\text{encoding the order } r) = 18$ bits are required to encode the Voronoi vector.

The above example shows that, some large vector components may consume more bits in STE compared with VE due to the higher extension order, but if the remaining components are small and require low-order or no extension, the total bits consumed by the vector can be saved. In general cases, our method is able to achieve similar coding efficiency compared with VE.

C. STE Searching Algorithm

The searching algorithm employed by STE aims to quickly find the $\mathbf{s}$, $\mathbf{r}$ and $\mathbf{m}$ for $\mathbf{y}$ to make $\mathbf{c}$ a base codebook vector, which is illustrated as follows.

TABLE III
COMPUTATIONAL COMPLEXITY/AVERAGE BITS ALLOCATION/MSE/SNR/ MEMORY REQUIREMENTS USING STE AND VE

	VE	STE
average operations per vector	add: 112 times shift: 82 times	add: 34 times shift: 0 times
Leader	37	23
Memory(Bytes)	1132	904
bits/vector	22.941	22.885
MSE	0.31	0.34
SNR	20.38	20.12

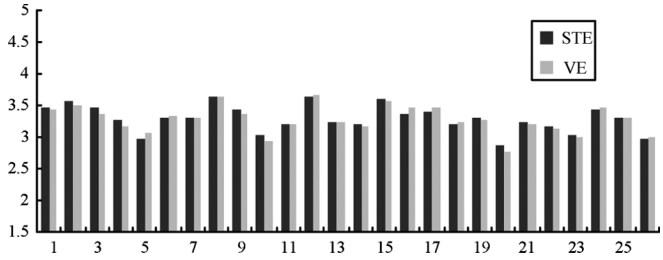

Fig. 2. Comparison of PESQ score at 24.0 kbps of 26 test sequences between AMR-WB+ with VE and STE.

1) Begin: Set $|s_i| = 0$ and $c_i = y_i$ for $i = 1, 2, \ldots, 8$. Also set the initial extension order and index for each split value: $r_i = 0, m_i = 0$.
2) If $\mathbf{c}$ is in the base codebook defined in Table I, stop the iteration, encode $\mathbf{c}$ by its base codebook index and encode s_i by r_i and m_i for $i = 1, 2, \ldots, 8$. Otherwise go to 3).
3) Find all i of $c_i > 4$. Increase corresponding $|s_i|$ by 4. If the increased $|s_i| < 2^{r_i+2}$, increase m_i by 1, otherwise increase r_i by 1 and set $m_i = 0$.
4) Calculate the new $\mathbf{c}$ by (7) and go to 2).

For example, assume that $\mathbf{y} = \{11, -1, -7, 1, 1, -1, 7, 1\}$. The STE iteration generates a series of $|\mathbf{s}|$, i.e., $\{4, 0, 4, 0, 0, 0, 4, 0\}$, $\{8, 0, 4, 0, 0, 0, 4, 0\}$, a series of $\mathbf{r}$, i.e., $\{1, 0, 1, 0, 0, 0, 1, 0\}$, $\{2, 0, 1, 0, 0, 0, 1, 0\}$ and a series of $\mathbf{m}$, i.e., $\{0, 0, 0, 0, 0, 0, 0, 0\}$, $\{0, 0, 0, 0, 0, 0, 0, 0\}$. Finally $\mathbf{c} = \{3, -1, -3, 1, 1, -1, 3, 1\}$ which is in the base codebook.

In this algorithm, only addition and subtraction operations are required to determine $\mathbf{s}$ and $\mathbf{c}$, which require ultra-low computational complexity.

IV. PERFORMANCE EVALUATION OF STE

We compare STE with VE to evaluate its performance in three aspects: computational complexity, storage requirement and quality.

Experiments are based on MPEG2-standardized testing sequences selected outside the training database, including three men's voices, three women's voices, seven music, and 13 symphony sequences sampled at 16 kHz. We incorporate the STE to TCX codec of AMR-WB+ frame to replace the original VE scheme. In this frame, the input sequences are first transformed to FFT domain and then quantized and encoded by VE-based or STE-based LVQ [7].

We evaluate the computational complexity of VE and STE in the term of the operations they employ. The matrix multiplication operations in calculating the Voronoi vector can be measured in the term of add and shift operations. Our STE scheme only employs add operations. The second row of Table III gives the average times employed by STE and VE to encode per vector in AMR-WB+ frame. The result shows that STE has ultra-low computational complexity compared to VE.

The third row of Table III indicates STE has fewer leaders in the base codebook compared with VE. Therefore, STE also uses less memory space than VE as shown in the fourth row of Table III.

For the coding quality evaluation, we first compare the average bits required by VE and STE to encode per vector. The result is shown in the fifth row of Table III. The sixth and the seventh row of Table III compare the MSE and SNR respectively between AMR-WB+ with VE and STE. Fig. 2 shows the PESQ scores for AMR-WB+ with VE and STE. The experiments illustrate that STE achieves similar coding quality compared with VE.

V. CONCLUSION

We present a practical Split Table Extension scheme aiming at resolving the problem of overload distortion in EAVQ frame. The STE scheme represents an overloaded lattice vector by the sum of the base vector and the split vector. Bit allocation for the split vector depends on the extension order of each vector component and can be highly flexible. The algorithm of searching the extension order and the split vector has ultra-low computational complexity. Experiments of applying STE to low-bitrate audio coding frame show that the presented scheme greatly reduces the computational complexity and storage requirement while having similar coding quality compared with Voronoi extension scheme.

REFERENCES

[1] D. G. Jeong and J. D. Gibbson, "Lattice vector quantization for image coding," in *ICASSP-89*, 1989, vol. 3, pp. 1743–1746.
[2] L. H. Fonteles and M. Antonini, "Lattice vector quantization for normal mesh geometry coding," in *ICASSP-06*, 2006, vol. 2, pp. 513–516.
[3] A. Vasuki and P. T. Vanathi, "A review of vector quantization techniques," *IEEE Potentials,* , vol. 25, pp. 39–47, 2006.
[4] M. Xie and J. P. Adoul, "Embedded algebraic vector quantization (EAVQ) with application to wideband speech coding," in *ICASSP-96*, 1996, vol. 1, pp. 240–243.
[5] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2325–2383, 1998.
[6] S. Ragot, B. Bessette, and R. Lefebvre, "Low-complexity multi-rate lattice vector quantization with application to wideband TCX speech coding at 32 kbit/s," in *ICASSP-04*, 2004, vol. 1, pp. 510–514.
[7] 3GPP-26290, Extended Adaptive Multi-Rate—Wideband (AMR-WB+) Codec: Transcoding Functions 3GPP Organization, 2005.
[8] J. H. Conway and N. J. A. Sloane, "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Trans. Inform. Theory*, vol. 28, pp. 227–232, 1982.
[9] B. Bessetee, S. Ragot, and J. P. Adoul, "Method and System for Multi-Rate Lattice Vector Quantization of a Signal," U.S. Patent Appl. 2005/0285764 A1, 2005.