Sensors and Materials, Vol. 31, No. 1 (2019) 153–163
MYU Tokyo

S & M 1757

High-efficiency Vector Quantization Codebook Search Algorithms for Extended Adaptive Multi-rate-wideband Audio Coder

Cheng-Yu Yeh and Hsiang-Yueh Lai*

Department of Electrical Engineering, National Chin-Yi University of Technology,
No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 41170, Taiwan (R.O.C)

(Received May 22, 2018; accepted September 5, 2018)

Keywords: audio codec, vector quantization (VQ), immittance spectral frequency (ISF), triangular inequality elimination (TIE), equal-average nearest neighbor search (ENNS)

The extended adaptive multi-rate-wideband (AMR-WB+) is a standard audio codec stipulated by the 3rd Generation Partnership Project (3GPP). It has been experimentally validated by 3GPP that an AMR-WB+ audio codec well outperforms others when encoding speech or audio signals at bitrates below 24 kbps, but at the cost of high computational load, particularly when performing the vector quantization (VQ) of immittance spectral frequency (ISF) coefficients. For this sake, by no means can an AMR-WB+ audio codec implemented on mobile devices meet the energy efficiency requirement. As a solution to this problem, in this paper, we present two high-efficiency triangular inequality elimination (TIE)-based algorithms for VQ codebook search. The first is referred to as the iterative TIE (ITIE) search algorithm, and the other is substantially the combined use of ITIE and the equal-average nearest neighbor search (ENNS) algorithm, both of which are subsequently applied to quantize the vectors of ISF coefficients. Most of the codewords are ruled out herein as the candidates using one-level and two-level rejection criteria, leading to a significantly reduced number of codeword searches but still maintaining the same coding quality. With a full search algorithm as a benchmark, this work shows a reduction in the total number of operations of more than 75.21%, a figure far beyond 60.23% using the TIE algorithm with a dynamic and an intersection mechanism (DI-TIE), 60.85% using ENNS, and 64.88% using the equal-average equal-variance equal-norm nearest neighbor search (EEENNS) algorithm. Moreover, this work can be implemented on handheld devices equipped with multiple sensing components, e.g., smartphones, to provide various multimedia and audio applications, and the energy saving requirement can be achieved as well.

1. Introduction

The extended adaptive multi-rate-wideband (AMR-WB+) is a standard audio codec stipulated by the 3rd Generation Partnership Project (3GPP)(1,2) and is substantially an extended version of the adaptive multi-rate-wideband (AMR-WB),(3,4) as its name indicates. Similarly to AMR-WB, AMR-WB+ works in the speech codec mode and is also designed to code and

*Corresponding author: e-mail: anne@ncut.edu.tw
https://doi.org/10.18494/SAM.2019.1998

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

decode stereo and audio signals sampled at up to 48 kHz. As suggested in 3GPP TR 26.936(5) on AMR-WB+ and Enhanced aacPlus (Eaac+)(6) audio codecs, an AMR-WB+ encoder has been validated to well perform in general when encoding speech or audio signals at bitrates below 24 kbps. Accordingly, AMR-WB+ is seen as an advantageous candidate when implemented on mobile devices.

As explicitly stated in Ref. 1, an AMR-WB+ encoder takes a mono or stereo signal sampled at 16–48 kHz and outputs a bit stream at bitrates between 6–48 kbps. This is because audio signals can be well encoded using the hybrid algebraic code-excited linear-prediction (ACELP)/transform coded excitation (TCX) technique, meaning that the encoder must be smart enough to identify the type of input signals and then choose the right coding technique, so as to well maintain the signal decoding quality. As indicated in TR 26.936,(5) however, an extremely high computational complexity occurs in the encoding procedure of AMR-WB+. In short, an AMR-WB+ codec cannot meet the energy saving requirement for mobile devices.

As described in Ref. 1, an AMR-WB+ encoder mainly involves 3 coding units, that is, a low-frequency unit using ACELP/TCX encoding, a high-frequency unit using bandwidth extension (BWE) encoding, and a stereo coding unit. These involve an enormous computational load to perform the vector quantization (VQ) of immittance spectral frequency (ISF) coefficients.(7–9) This is simply for the reason that the VQ of ISF coefficients is required in both the high-frequency coding unit and particularly the low-frequency one, where VQ of ISF is performed up to 7 times in each frame, which possess the same VQ structure as an AMR-WB encoder. A combination of the split VQ (SVQ) and multistage VQ (MSVQ) techniques, referred to as the split-multistage VQ (S-MSVQ), is employed therein to quantize the ISF coefficients. Conventionally, VQ conducts a full search to ensure that a codeword best matches an arbitrary input vector, but the full search requires an enormous computational load.

Over the years, much effort has been made to address the issue of computational load reduction for VQ codebook search, including those based on the equal-average equal-variance equal-norm nearest neighbor search (EEENNS) algorithms(10–14) and the triangular inequality elimination (TIE) algorithms.(15–18) The EEENNS algorithm, derived from the equal-average nearest neighbor search (ENNS) and the equal-average equal-variance nearest neighbor search (EENNS) approaches, uses three significant features of a vector, i.e., the mean value, the variance, and the norm, as a three-level elimination criterion to reject impossible codewords. Thus, the aim of the computational load reduction can be reached.

As presented in Ref. 18, the TIE algorithm with a dynamic and an intersection mechanism, abbreviated as DI-TIE, was well validated to outperform the multiple TIE (MTIE) algorithm and EEENNS. Nevertheless, there was hidden computational cost in DI-TIE when ruling out codewords as candidates. As will be illustrated later, the computational complexity reduction in Ref. 18 is not as much as it seems to be. On the other hand, a binary search space-structured VQ (BSS-VQ) algorithm is adopted to reduce the complexity of ISF quantization in AMR-WB.(19) In this manner, a trade-off can be made between the quantization accuracy and the search load to meet a user's need when performing a VQ encoding. However, this coding quality is expected to decline.

To prevent this decline, in this paper, we present two improved versions of TIE-based algorithms to further improve the codebook search performance when quantizing ISF vectors. The first is referred to as the iterative TIE (ITIE) algorithm, and the second is substantially a combined use of ITIE and ENNS, both of which are expected to remarkably enhance the coding efficiency of an AMR-WB+ audio codec, namely, reduce the computational complexity of AMR-WB+ encoding. The codebook search performance of this proposal is tested in S-MSVQ of ACELP/TCX encoding. For comparison purposes, the search performance of the algorithms herein is assessed in terms of the total number of operations, that is, addition + subtraction, multiplication + division, comparison, and square root. The presented algorithms are ultimately validated to well outperform their counterparts, and are expected to meet the energy efficiency requirement when implemented on an AMR-WB+ codec of mobile devices. In addition, the applications of audio/voice communication can be provided on smartphones by means of audio sensing components and the AMR-WB+ audio codec.

This paper is outlined as follows. The ISF coefficient quantization in AMR-WB+ is described in Sect. 2. In Sect. 3, we present two search algorithms for ISF quantization. Experimental results are demonstrated and discussed in Sect. 4. This work is summarized at the end of this paper.

2. Procedure of ISF Quantization in ACELP/TCX Encoding

In AMR-WB+, a linear prediction analysis is made as follows. As the first step, a frame is applied to evaluate linear predictive coefficients (LPCs), which are then converted into ISF coefficients. Subsequently, quantized ISF coefficients are obtained following a VQ process, which is detailed below.

2.1 Linear prediction analysis

In a linear prediction, a Levinson–Durbin algorithm is used to compute the 16th-order LPC, a_i, of a linear prediction filter, defined as

$$\frac{1}{A(z)} = \frac{1}{1+\sum_{i=1}^{16} a_i z^{-i}}. \tag{1}$$

Subsequently, the LPC parameters are converted into the immittance spectral pair (ISP) coefficients for the purposes of parametric quantization and interpolation. The ISP coefficients are defined as the roots of the following two polynomials.

$$F_1'(z) = A(z) + z^{-16}A(z^{-1}) \tag{2}$$

$$F_2'(z) = A(z) - z^{-16}A(z^{-1}) \tag{3}$$

$F_1'(z)$ and $F_2'(z)$ are symmetric and antisymmetric polynomials, respectively. It can be proven that all the roots of such two polynomials lie and alternate successively on a unit circle in the z-domain. Also, $F_2'(z)$ has two roots at $z = 1$ ($\omega = 0$) and $z = -1$ ($\omega = \pi$). Such two roots are eliminated by introducing the following polynomials, with eight and seven conjugate roots on the unit circle expressed respectively as

$$F_1(z) = F_1'(z) = (1 + a_{16}) \prod_{i=0,2,\ldots,14} (1 - 2q_i z^{-1} + z^{-2}), \tag{4}$$

$$F_2(z) = \frac{F_2'(z)}{(1 - z^{-2})} = (1 - a_{16}) \prod_{i=1,3,\ldots,13} (1 - 2q_i z^{-1} + z^{-2}), \tag{5}$$

where the coefficients q_i are referred to as the ISPs in the cosine domain and a_{16} is the last predictor coefficient. A Chebyshev polynomial is used to solve Eqs. (4) and (5). Finally, derived from the ISP coefficients, 16th-order ISF coefficients ω_i can be obtained by taking the transformation $\omega_i = \arccos(q_i)$.

2.2 Quantization of ISF coefficients

Before a quantization process, the mean-removed and first-order moving average (MA) filtering process are performed on the ISF coefficients to obtain a residual ISF vector,(1,3) that is,

$$\boldsymbol{r}(n) = \boldsymbol{z}(n) - \boldsymbol{p}(n), \tag{6}$$

where $\boldsymbol{z}(n)$ and $\boldsymbol{p}(n)$ respectively denote the mean-removed and predicted ISF vector at frame n by a first-order MA prediction,(1,3) defined as

$$\boldsymbol{p}(n) = \frac{1}{3}\hat{\boldsymbol{r}}(n-1), \tag{7}$$

where $\hat{\boldsymbol{r}}(n-1)$ is the quantized residual vector at the previous frame.

Subsequently, S-MSVQ is performed on $\boldsymbol{r}(n)$. The structure of S-MSVQ is presented in Table 1. In stage 1, $\boldsymbol{r}(n)$ is split into two subvectors, namely, a 9-dimensional subvector $\boldsymbol{r}_1(n)$ associated with codebook CB1 and a 7-dimensional subvector $\boldsymbol{r}_2(n)$ associated with codebook CB2, for VQ encoding. As a preliminary step of stage 2, the quantization error vectors are split into five subvectors, symbolized as $\boldsymbol{r}_i^{(2)} = \boldsymbol{r}_i - \hat{\boldsymbol{r}}_i,\ i = 1,2$. For instance, $\boldsymbol{r}^{(2)}{}_{1,1\text{–}3}$ in Table 1 represents the subvector split from the 1st to the 3rd components of $\boldsymbol{r}_1$, and then VQ encoding is performed thereon over codebook CB11 in stage 2. Likewise, $\boldsymbol{r}^{(2)}{}_{2,4\text{–}7}$ stands for the subvector split from the 4th to the 7th components of $\boldsymbol{r}_2$, after which VQ encoding is performed over codebook CB22 in stage 2. Finally, a squared error ISF distortion measure, that is, Euclidean distance, is used in all the quantization processes.

Table 1
Structure of S-MSVQ in ACELP/TCX encoding.

Stage 1	CB1: $\boldsymbol{r}_1$ (1–9 order of $\boldsymbol{r}$) (8 bits)			CB2: $\boldsymbol{r}_2$ (10–16 order of $\boldsymbol{r}$) (8 bits)	
Stage 2	CB11: $\boldsymbol{r}^{(2)}{}_{1,1\text{–}3}$ (6 bits)	CB12: $\boldsymbol{r}^{(2)}{}_{1,4\text{–}6}$ (7 bits)	CB13: $\boldsymbol{r}^{(2)}{}_{1,7\text{–}9}$ (7 bits)	CB21: $\boldsymbol{r}^{(2)}{}_{2,1\text{–}3}$ (5 bits)	CB22: $\boldsymbol{r}^{(2)}{}_{2,4\text{–}7}$ (5 bits)

3. Proposed Search Algorithms

In this paper, we present two search algorithms, the first of which is the ITIE algorithm and the second is the combined use of ITIE and ENNS, symbolized as ITIE + ENNS, read as ITIE *plus* ENNS and designated as the ITIEpENNS search algorithm. As a preliminary to the ITIE algorithm, TIE and DI-TIE are briefly described as follows.

A TIE algorithm works as follows. As presented in Ref. 15, a codeword captured from the preceding frame is treated as the reference codeword $\boldsymbol{c}_r$ in the current frame. Subsequently, the Euclidean distance between the input vector $\boldsymbol{x}$ and $\boldsymbol{c}_r$, symbolized as $d(\boldsymbol{c}_r, \boldsymbol{x})$, is evaluated. A set composed of all the codewords $\boldsymbol{c}_i$ satisfying the condition $d(\boldsymbol{c}_r, \boldsymbol{c}_i) < 2d(\boldsymbol{c}_r, \boldsymbol{x})$ is referred to as a candidate search group (CSG), denoted by CSG($\boldsymbol{c}_r$) and formulated as

$$\text{TIE: } CSG(\boldsymbol{c}_r) = \{\boldsymbol{c}_i \mid d(\boldsymbol{c}_r, \boldsymbol{c}_i) < 2d(\boldsymbol{c}_r, \boldsymbol{x})\},\ 1 \le i \le CNum\ , \tag{8}$$

where *CNum* represents the total number of codewords. CSG($\boldsymbol{c}_r$) is the search space over which a current codebook search is performed. In addition, a lookup table listing all the codewords sorted by the Euclidean distance is prebuilt for the TIE search algorithm.

As stated explicitly previously, the mean-removed and first-order MA filtering processes are performed on the ISF coefficients in AMR-WB+ to obtain a residual ISF vector $\boldsymbol{r}(n)$ before a quantization process. Consequently, the correlation between neighboring input vectors is gone, meaning that the TIE algorithm has a poor performance when applied to a codebook search.

As shown in Algorithm 1 for readers' convenience, the DI-TIE algorithm, suggested in Ref. 18, works in a different way. That is, a dynamic mechanism is enabled during a TIE codebook search. A CSG is automatically updated once a condition is met, and the intersection of the updated CSG and the previous one is found as a way to reduce the number of candidate codewords, denoted by $N(\boldsymbol{c}_r)$. The aim of DI-TIE is then to reduce the value of $N(\boldsymbol{c}_r)$ as much as it can during codebook search.

```
Algorithm 1: Search procedure of DI-TIE
Step 1:  Build a TIE lookup table.
Step 2:  Compute the $d(\boldsymbol{c}_r, \boldsymbol{x})$, and then determine the initial search space according to Eq. (8),
         that is,
```

$$CSG(\boldsymbol{c}_r) = \{\boldsymbol{c}_k \mid k = 1, 2, \ldots, N(\boldsymbol{c}_r)\}\,, \tag{9}$$

where $\boldsymbol{c}_k$ and $N(\boldsymbol{c}_r)$ denote the codewords and the number thereof in $CSG(\boldsymbol{c}_r)$, respectively.

Step 3: Starting at $k = 1$, assign $\boldsymbol{c}_k$ as a new reference codeword, compute $d(\boldsymbol{c}_k, \boldsymbol{x})$, and then determine $CSG(\boldsymbol{c}_k)$ and $N(\boldsymbol{c}_k)$.

Step 4: If $N(\boldsymbol{c}_k) < N(\boldsymbol{c}_r) - k$, then perform the set intersection operation in Eq. (10), update $N(\boldsymbol{c}_r)$, let $k = 1$, and repeat Step 3.

$$CSG(\boldsymbol{c}_r) = CSG(\boldsymbol{c}_k) \cap CSG(\boldsymbol{c}_r) \tag{10}$$

Otherwise, let $k = k + 1$, and then repeat Steps 3 and 4, until $k = N(\boldsymbol{c}_r)$.

Even though DI-TIE does as it claims to significantly reduce the number of codeword searches, there is a hidden computational cost, i.e., the operations in Steps 3 and 4. In short, the overall computational complexity reduction is not as much as it seems to be. As a solution to this issue, two high-efficiency search algorithms are developed and detailed as follows.

3.1 ITIE search algorithm

As illustrated in Fig. 1 and stated in Algorithm 2, codeword captured from the preceding frame is treated as the reference codeword $\boldsymbol{c}_r$ in the current frame, just as in the TIE case. A codebook search is performed over $CSG(\boldsymbol{c}_r)$ using Eq. (8). As in the TIE and DI-TIE cases, the condition $d(\boldsymbol{c}_r, \boldsymbol{c}_k) < 2d(\boldsymbol{c}_r, \boldsymbol{x})$ is checked. If the condition is true, then compute $d(\boldsymbol{c}_k, \boldsymbol{x})$, and check another condition $d(\boldsymbol{c}_k, \boldsymbol{x}) < d(\boldsymbol{c}_r, \boldsymbol{x})$. If the condition is true, then $\boldsymbol{c}_r$ is replaced with $\boldsymbol{c}_k$, and the search scope $CSG(\boldsymbol{c}_r)$ is updated. $CSG(\boldsymbol{c}_r)$ updates and shrinks each time through the above-stated loop. In this manner, ITIE requires a smaller number of codeword searches but at the cost of the additional operations through the loop, and turns out to well outperform the TIE and DI-TIE counterparts with respect to the overall computational complexity.

Algorithm 2: Search procedure of ITIE

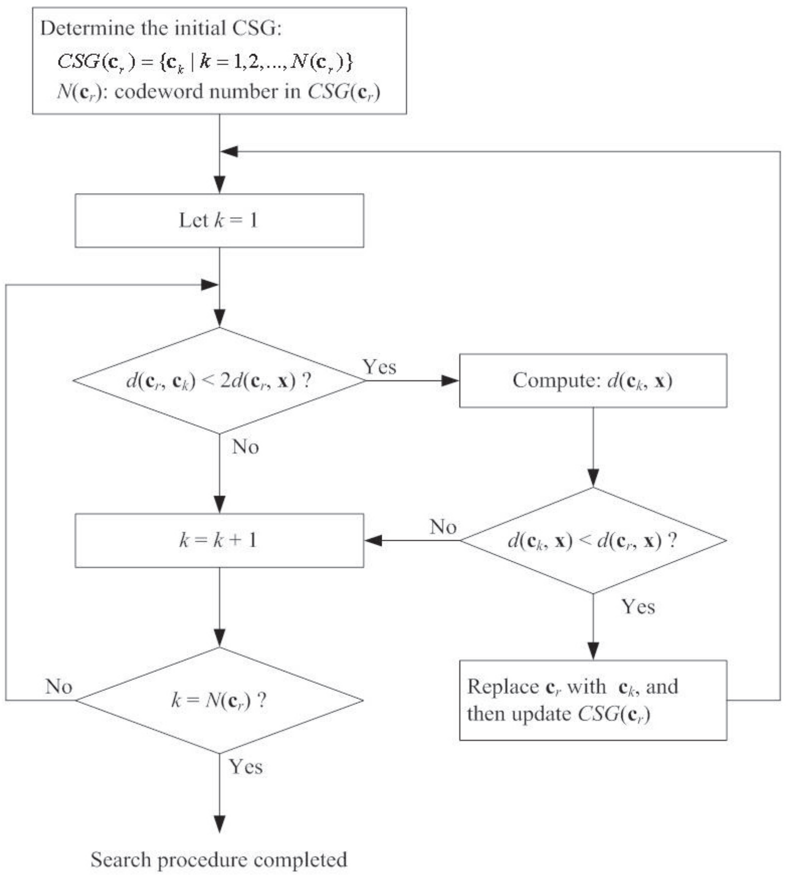
Step 1: Build a TIE lookup table.

Step 2: Given a $\boldsymbol{c}_r$, compute $d(\boldsymbol{c}_r, \boldsymbol{x})$, and then $CSG(\boldsymbol{c}_r)$ is found directly in the TIE lookup table.

Step 3: Starting at $k = 1$, then obtain $d(\boldsymbol{c}_r, \boldsymbol{c}_\mathrm{k})$ from the lookup table.

Step 4: If $d(\boldsymbol{c}_r, \boldsymbol{c}_\mathrm{k}) < 2d(\boldsymbol{c}_r, \boldsymbol{x})$, then compute $d(\boldsymbol{c}_\mathrm{k}, \boldsymbol{x})$, and then perform Step 5.
Otherwise, let $k = k + 1$, and then repeat Step 4, until $k = N(\boldsymbol{c}_r)$.

Step 5: If $d(\boldsymbol{c}_k, \boldsymbol{x}) < d(\boldsymbol{c}_r, \boldsymbol{x})$, then replace $\boldsymbol{c}_r$ with $\boldsymbol{c}_k$, update new $CSG(\boldsymbol{c}_r)$, let $k = 1$, and repeat Step 3.
Otherwise, let $k = k + 1$ and then repeat Step 4 until $k = N(\boldsymbol{c}_r)$.



Determine the initial CSG:
$CSG(\mathbf{c}_r) = \{\mathbf{c}_k \mid k = 1,2,\ldots,N(\mathbf{c}_r)\}$
$N(\mathbf{c}_r)$: codeword number in $CSG(\mathbf{c}_r)$
Let $k = 1$
$d(\mathbf{c}_r, \mathbf{c}_k) < 2d(\mathbf{c}_r, \mathbf{x})$?
Yes
No
Compute: $d(\mathbf{c}_k, \mathbf{x})$
$d(\mathbf{c}_k, \mathbf{x}) < d(\mathbf{c}_r, \mathbf{x})$?
No
Yes
$k = k + 1$
$k = N(\mathbf{c}_r)$?
No
Yes
Replace $\mathbf{c}_r$ with $\mathbf{c}_k$, and then update $CSG(\mathbf{c}_r)$
Search procedure completed

Fig. 1. Flowchart of the ITIE search approach.

3.2 ITIEpENNS search algorithm

The ITIEpENNS algorithm is developed as a way to further reduce the overall computational complexity. First, the ENNS search algorithm is briefly described as follows.

Equal average is treated as a requirement to filter candidate codewords in ENNS using

$$m_x - \frac{d_{min}}{\sqrt{Dim}} < m_i < m_x + \frac{d_{min}}{\sqrt{Dim}}, \tag{11}$$

$$m_i = \frac{1}{Dim}\sum_{k=1}^{Dim} \boldsymbol{c}_i(k), \tag{12}$$

$$m_x = \frac{1}{Dim}\sum_{k=1}^{Dim} \boldsymbol{x}(k), \tag{13}$$

where d_{min} represents the shortest Euclidean distance so far, Dim denotes the dimension of vectors, and m_i and m_x symbolize the average of all the elements contained in the codeword $\boldsymbol{c}_i$ and the input vector $\boldsymbol{x}$, respectively.

If the condition in Eq. (11) is met, $\boldsymbol{c}_i$ is likely to be the best codeword, and $d(\boldsymbol{c}_i, \boldsymbol{x})$ is evaluated subsequently. Otherwise, there is a zero possibility that $\boldsymbol{c}_i$ can be the best codeword, and certainly, there is no need to compute $d(\boldsymbol{c}_i, \boldsymbol{x})$. For the sake of computational complexity reduction, a prebuilt lookup table is constructed so as to rapidly reference m_i. As a result, Eq. (11) merely takes a square root, 2 additions, those for m_x in Eq. (13) and a smaller number of compare operations. In this work, ITIE is integrated with ENNS, simply for the reason that it requires a smaller number of operations than EENNS and EEENNS.

The ITIEpENNS search algorithm is illustrated as a flowchart in Fig. 2. As compared with Fig. 1, $d(\boldsymbol{c}_k, \boldsymbol{x})$ is evaluated in Fig. 2 only if the condition $d(\boldsymbol{c}_r, \boldsymbol{c}_k) < 2d(\boldsymbol{c}_r, \boldsymbol{x})$ in ITIE and the other condition, given in Eq. (11), are met.

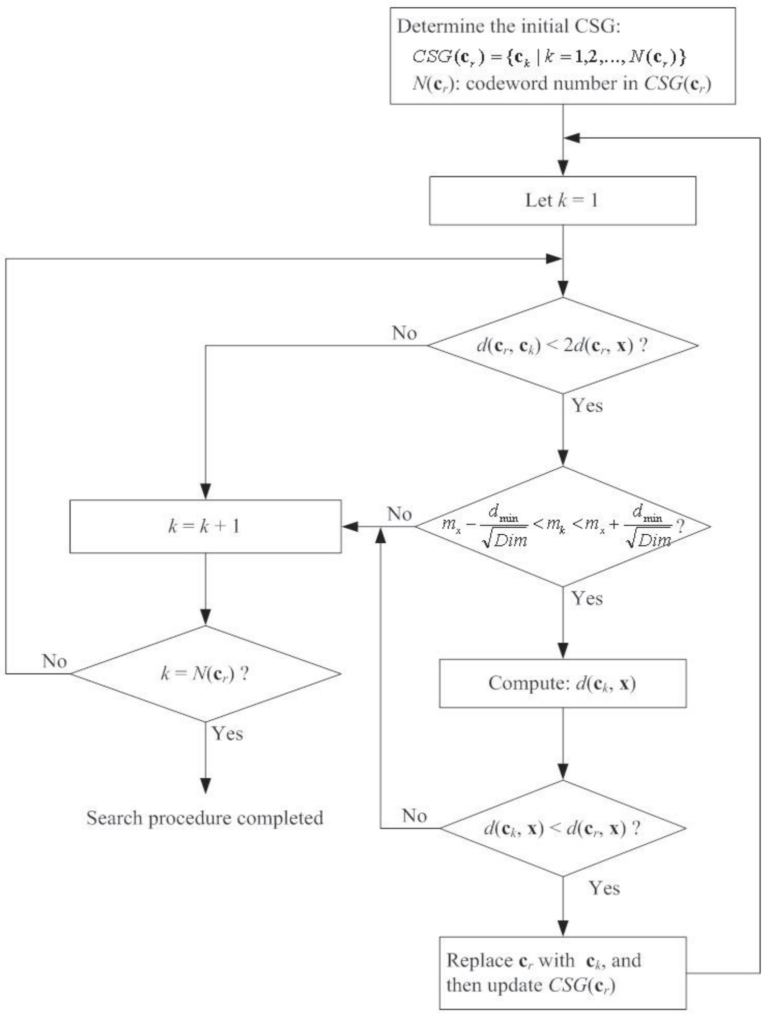

Determine the initial CSG:
$CSG(\mathbf{c}_r) = \{\mathbf{c}_k \mid k = 1,2,\ldots,N(\mathbf{c}_r)\}$
$N(\mathbf{c}_r)$: codeword number in $CSG(\mathbf{c}_r)$
Let $k = 1$
$d(\mathbf{c}_r, \mathbf{c}_k) < 2d(\mathbf{c}_r, \mathbf{x})$?
No
Yes
$m_x - \frac{d_{\min}}{\sqrt{Dim}} < m_k < m_x + \frac{d_{\min}}{\sqrt{Dim}}$?
No
Yes
$k = k + 1$
$k = N(\mathbf{c}_r)$?
No
Yes
Search procedure completed
Compute: $d(\mathbf{c}_k, \mathbf{x})$
$d(\mathbf{c}_k, \mathbf{x}) < d(\mathbf{c}_r, \mathbf{x})$?
No
Yes
Replace $\mathbf{c}_r$ with $\mathbf{c}_k$, and then update $CSG(\mathbf{c}_r)$

Fig. 2. Flowchart of the ITIEpENNS search approach.

4. Experimental Results

Performance is compared among the presented ITIE and ITIEpENNS, a DI-TIE, an ENNS, and an EEENNS. Since TIE-based and ENNS-based VQ codebook searches are conducted, a 100% search accuracy is obtained by such five algorithms in comparison with a full search approach. For this sake, performance is compared in terms of computational complexity. The test objects are those selected from a speech database with one male and one female speaker, containing 600 sentences for a duration of over 121 min and 363,012 frames.

Firstly, Table 2 gives the average number of operations for various search algorithms, i.e., addition + subtraction, denoted by Add, multiplication + division, denoted by Mul, compare, denoted by Comp, and square root, denoted by Sqrt in CB1 and CB2, i.e., stage 1 of S-MSVQ, codebook search. For comparison purposes, Table 3 gives the average number of respective total operations required in a complete S-MSVQ codebook search, namely, a search over all the codebooks.

As indicated in Tables 2 and 3, a codebook search requires a significantly larger number of operations in stage 1 than in stage 2. For instance, the full search algorithm in stage 1 performs 6912 (CB1) + 5376 (CB2) = 12288 out of 15840 operations, the total required in a complete S-MSVQ codebook search, meaning that the computational complexity is dominated by stage 1. As such, a way must be found to significantly cut the number of operations in stage 1 as the key to a notable overall computational complexity reduction.

Table 2
Comparison of the average numbers of respective operations over the CB1 and CB2 codebook searches among various algorithms.

Codebook	Method	Operations				
		Add	Mul	Comp	Sqrt	Total
CB1	Full search	4352.00	2304.00	256.00	0.00	6912.00
	DI-TIE	982.87	424.60	721.98	0.00	2129.45
	ENNS	1298.75	684.15	330.53	3.38	2316.81
	EEENNS	1037.13	545.72	448.62	8.76	2040.23
	ITIE	986.07	525.88	173.85	0.00	1685.80
	ITIEpENNS	750.24	397.56	216.05	3.84	1367.69
CB2	Full search	3328.00	1792.00	256.00	0.00	5376.00
	DI-TIE	821.77	342.52	852.47	0.00	2016.76
	ENNS	1128.13	604.87	340.59	4.72	2078.31
	EEENNS	871.62	462.80	469.77	11.44	1815.63
	ITIE	806.45	439.19	185.95	0.00	1431.59
	ITIEpENNS	656.81	356.00	234.25	4.94	1252.00

Table 3
Comparison of the averages of the total numbers of operations over a complete S-MSVQ codebook search among various algorithms.

Method	Operations				
	Add	Mul	Comp	Sqrt	Total
Full search	9664.00	5280.00	896.00	0.00	15840.00
DI-TIE	2439.42	1025.72	2835.17	0.00	6300.31
ENNS	3223.68	1756.66	1189.62	30.85	6200.81
EEENNS	2540.21	1344.54	1603.22	75.69	5563.66
ITIE	2146.47	1198.96	580.58	0.00	3926.01
ITIEpENNS	1744.31	971.56	719.64	32.96	3468.47

As also revealed in Table 3, the full search algorithm ranks first with respect to the total number of operations among the first 4 search algorithms, and it is as well noted that the remaining two, the presented ITIE and ITIEpENNS, outperform the others to a great extent as intended. A close observation also reveals that DI-TIE, ENNS, and EEENNS require the greatest number of Comp, Add/Mul, and Sqrt operations, respectively. In contrast, ITIE and ITIEpENNS turn out to have the smallest number of Comp/Sqrt and Add/Mul operations, respectively.

Eventually, Table 4 gives a comparison on the total number of operations required by a complete S-MSVQ codebook search among various algorithms. With the full search algorithm as a benchmark, a high computational complexity reduction is reflected by a high value of the computational saving (CS), and CS is tabulated in the last row of Table 4 and illustrated as a bar graph in Fig. 3. As can be seen in Fig. 3, ITIEpENNS with a CS of 78.10% and ITIE with 75.21% rank first and second, respectively, and are experimentally validated to well outperform the counterparts, EEENNS with 64.88%, ENNS with 60.85%, and DI-TIE with 60.23%.

Table 4
Overall performance comparison among various methods.

Method		Full search	DI-TIE	ENNS	EEENNS	ITIE	ITIEpENNS
Stage 1	CB1	6912.00	2129.45	2316.81	2040.23	1685.80	1367.69
	CB2	5376.00	2016.76	2078.31	1815.63	1431.59	1252.00
Stage 2	CB11	576.00	386.88	297.36	280.72	154.76	163.04
	CB12	1152.00	627.93	517.88	459.17	179.82	191.74
	CB13	1152.00	578.30	493.32	433.47	169.10	180.53
	CB21	288.00	221.78	189.40	203.39	111.51	120.62
	CB22	384.00	339.21	307.73	331.05	193.43	192.85
Overall		15840.00	6300.31	6200.81	5563.66	3926.01	3468.47
CS (%)		benchmark	60.23	60.85	64.88	75.21	78.10

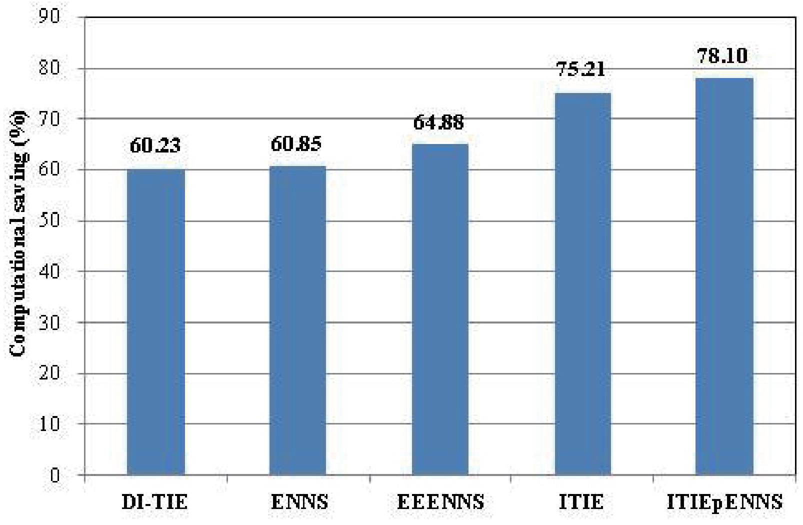


Fig. 3. (Color online) A bar graph representation of Table 4.

5. Conclusions

In this paper, we present two improved versions of TIE-based algorithms, namely, ITIE and ITIEpENNS, for VQ codebook search, and both are then applied to ISF coefficient quantization in an AMR-WB+ audio codec. A multitude of codewords are ruled out as candidates using a one-level rejection criterion in ITIE and a two-level in ITIEpENNS, giving rise to a much smaller number of codeword searches, i.e., a significant computational complexity reduction but with the same coding quality. The improved versions are validated by experiments to well outperform the DI-TIE, ENNS, and EEENNS counterparts. Furthermore, the improved AMR-WB+ audio codec can be applied to mobile devices equipped with a variety of sensing elements, e.g., smartphones, to provide various multimedia and audio applications, and the energy saving requirement can be fulfilled as well.

Acknowledgments

This research was financially supported by the Ministry of Science and Technology under grant number MOST 106-2221-E-167-030, Taiwan. The authors are deeply indebted as well to the coauthors of a cited paper, B.J. Yao and S.H. Hwang, for their contribution as the basis of this work.

References

1 3GPP TS 26.290: Audio codec processing functions; Extended Adaptive Multi-Rate-Wideband (AMR-WB+) codec; Transcoding functions (3GPP, Valbonne, France, 2015).
2 R. Salami, R. Lefebvre, A. Lakaniemi, K. Kontola, S. Bruhn, and A. Taleb: IEEE Commun. Mag. **44** (2006) 90.
3 3GPP TS 26.190: Adaptive Multi-Rate-Wideband (AMR-WB) speech codec; Transcoding functions (3GPP, Valbonne, France, 2012).
4 B. Bessette, R. Salami, R. Lefebvre, M. Jelínek, J. Rotola-Pukkila, J. Vainio, H. Mikkola, and K. Järvinen: IEEE Trans. Speech Audio Process. **10** (2002) 620.
5 3GPP TR 26.936: Performance characterization of 3GPP audio codecs (3GPP, Valbonne, France, 2015).
6 3GPP TS 26.401: General audio codec audio processing functions; Enhanced aacPlus general audio codec; General description (3GPP, Valbonne, France, 2015).
7 W. J. Han, E. K. Kim, and Y. H. Oh: IEEE Signal Process. Lett. **9** (2002) 418.
8 S. Chatterjee and T. V. Sreenivas: Signal Process. **88** (2008) 1528.
9 C. Salah-Eddine and B. Merouane: Speech Commun. **65** (2014) 94.
10 Z. M. Lu and S. H. Sun: IEICE Trans. Inf. Syst. **E86-D** (2003) 660.
11 S. C. Chu, Z. M. Lu, and J. S. Pan: Inf. Sci. **177** (2007) 734.
12 S. X. Chen, F. W. Li, and W. L. Zhu: IET Image Process. **2** (2008) 275.
13 S. X. Chen and F. W. Li: IET Image Process. **5** (2011) 18.
14 S. Xia, Z. Xiong, Y. Luo, L. Dong, and G. Zhang: Knowledge-Based Syst. **90** (2015) 99.
15 S. H. Hwang and S. H. Chen: Electron. Lett. **26** (1990) 1618.
16 C. H. Hsieh and Y. J. Liu: IEEE Trans. Image Process. **9** (2000) 321.
17 C. P. Chu, C. Y. Yeh, and S. H. Hwang: IEEJ Trans. Electr. Electron. Eng. **9** (2014) S70.
18 B. J. Yao, C. Y. Yeh, and S. H. Hwang: Symmetry **8** (2016) 1.
19 C. Y. Yeh: Symmetry **9** (2017) 1.