Science Academy Transactions on Computer and Communication Network (SATCCN)
Vol. 1, No. 3, December 2011, ISSN: 2046-5157
© Science Academy Publisher, United Kingdom
www.sciacademypublisher.com

An Efficient Codebook Search Method for Speech Coders Optimized by Evolutionary and Swarm-Based Techniques

Mansour Sheikhan, Sahar Garoucy, and S. Amir Ghoreishi

Department of Electrical Engineering, Islamic Azad University, South Tehran Branch, Tehran, Iran

Abstract – The computational complexity in many signal processing systems is due to codebook search that is a time-consuming process. Finding efficient codebook search methods has attracted many research efforts in the recent years. In this paper, we suggest a search codebook method based on the magnitude behavior of inverse filtered target signal (MBIFTS) and intelligent optimization techniques. This method considers the correlation between target signal and impulse response with the aim of selecting the optimal positions corresponding to larger amplitudes. In this way, the position of maximum synthesis waveform is assumed as the best position in the search process. To improve the efficiency of this method, several optimization algorithms such as bee colony algorithm (BCA), genetic algorithm (GA), imperialist competition algorithm (ICA), differential evolution (DE) algorithm, and particle swarm optimization (PSO) algorithm are investigated. All of the mentioned optimization methods are employed in an adaptive multi-rate wideband (AMR-WB) speech coder. It should be noted that the proposed modifications can be used in all of the coders that are based on algebraic code excited linear prediction (ACELP) algorithm. The performance comparison of traditional implementation of ITU-T G.722.2 recommendation and proposed optimized codebook search methods shows that the BCA-optimized codebook search scheme performs better in terms of execution-time and reduction in the number of operations without significant degradation in the quality metrics.

Keywords – Fast codebook search, Intelligent optimization algorithms, ACELP, ICA, DE, BCA, PSO, GA

1. Introduction

The codebook search is based on selection of the best optimal excitation vector using a closed-loop mechanism to find the vector with minimum perceptually-weighted distortion in many speech coding algorithms such as code excited linear prediction (CELP) method. As a member of CELP-family coders, the algebraic CELP (ACELP) has a fixed algebraic codebook structure [1]. ACELP method is recommended in several speech coding recommendations and standards such as ITU-T G.723.1 [2], adaptive multi-rate (AMR) [3], adaptive multi-rate wideband (AMR-WB) [4, 5], extended adaptive multi-rate wideband (AMR-WB+) [6], and variable-rate multimode wideband (VMR-WB) [7]. Most of the complexity in these coders is due to the codebook search.

Because of the large number of candidate code-vectors, full search is not efficient. So, several methods have been proposed to improve the codebook search [8-13]. As example researches, Laflamme et al. [8] have used focused search on a portion of the algebraic codebook and Parvin Kumar [13] has proposed four methods to improve the codebook search: 1) depth first tree search (DFTS) method, 2) a pruning tree method, 3) maximum-take-precedence (MTP) method in which all the positions of each track is divided to several regions and the search process is performed only for some regions, 4) reordering search sequence (RSS) method in which the sequence of codebook search is reordered according to the mean square weighted error. Also, Halimi et al. [14] have proposed a multistage technique for CELP coder to reduce the time of codebook search using Trellis search.

On the other hand, several intelligent techniques (such as artificial neural networks) have been used in the speech processing systems e.g. speech recognition [15-18], speech synthesis [19, 20], speech coding [21-27], emotion recognition from speech [28, 29], and syntactic processing [30] that have been experienced by the authors. Also, evolutionary techniques such as genetic algorithm (GA) or particle swarm optimization (PSO) algorithm have been used in these systems to determine the optimal values for parameters of these systems [26-28].

The main idea in this study is to introduce a method for searching the codebook based on investigating the magnitude behavior of inverse filtered target signal (MBIFTS). In addition, to determine the optimum value for a parameter in the proposed method, several classic and modern optimization algorithms (such as GA, PSO, bee colony algorithm (BCA), differential evolution (DE) algorithm, and imperialist competition algorithm (ICA)) are used.

It is noted that there are many optimization algorithms which perform efficiently in the applications in which some parameters should be selected optimally. Obviously, we cannot determine which one is the best. It is depended on the problem that should be solved. For example, GA is a particular class of evolutionary algorithms that uses techniques inspired by evolutionary biology such as inheritance, mutation, and recombination [31]. GA finds approximate solutions to optimization and search problems. Also, PSO algorithm proposed by Kennedy and Eberhart [32] is motivated by social behavior of organisms. PSO provides a population-based search procedure in which individuals, called particles, change their position (state) with time. In PSO, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience and neighboring particle, making use of the best position encountered by itself and its neighbor. Thus, as in modern GAs, a PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation [33, 34]. In BCA, a colony of honey bees can be seen as a diffuse creature which can extend itself over long distances in multiple directions in order to exploit a large number of food sources at the same time as an optimization approach [35, 36]. Furthermore, DE algorithm has been presented by Storn and Price [37] as a heuristic optimization method which can be used to minimize nonlinear and non-differentiable continuous space functions with real-valued parameters [38]. Finally, ICA is a modern algorithm inspired from historical imperialistic competition among empires in socio-political world [39]. ICA is a global search algorithm that is based on that in real world different countries try to extend their power over other countries in order to use their resources and bolster their own government. This phenomenon is referred to as imperialism in social-political context. In fact, imperialist countries attempt to dominate other countries and turning them to their colonies by direct rules or by means of less obvious methods like controlling markets as it is more common today.

This paper is organized as follows. Section 2 gives an overview of codebook search structure in ACELP. In Section 3, the foundation of five optimization algorithms which are used in this study is reviewed. The proposed codebook search method, i.e. MBIFTS which is equipped with each of mentioned optimization methods, is introduced in Section 4. The experimental results are reported in Section 5. Finally, the paper is concluded in Section 6.

2. Codebook Search in ITU-T G.722.2 Recommendation

The aim of algebraic codebook search is to find the code-vector c_k which minimizes the square error E_k between the weighted target speech x and the reconstructed speech $\hat{x}$ as [5]:

$$E_k = \|x - \hat{x}\|^2 = \|x - g_c H c_k\|^2 \tag{1}$$

where k is the algebraic code-vector index, g_c denotes the codebook gain, and c_k is kth code-vector with size of L samples. L is the size of sub-frame according to the selected ACELP structure. Also, H is a lower triangular Toeplitz matrix ($L \times L$). Minimizing (1) is equivalent to maximizing the following equation [5]:

$$Q_k = \frac{(d^t c_k)^2}{c_k^t \varphi \, c_k} \tag{2}$$

where d is the inverse filtered target signal, as shown in (3), and ϕ is the correlation matrix of the impulse response $h(n)$.

$$d(n) = H^t x_2 = \sum_{i=n}^{L} x_2(i) h(i-n); n = 0, \ldots, L \tag{3}$$

In (2), c_k contains L samples, but most of them are zero pulses. The full search will be much complex since many combinations are possible as:

$${}_{L}C_{N_p} = \frac{L!}{(L-N_p)!N_p!} \tag{4}$$

where N_p is the number of nonzero pulses. So, a sub-optimal solution is generally used to find the code-vector. Since the algebraic code-vector contains few nonzero pulses, the numerator in (2) is given by [5]:

$$C = \sum_{i=0}^{Np-1} v_i d(m_i) \tag{5}$$

where m_i is the position of the ith pulse and v_i is the sign of pulse. Also, the denominator of (2) is given by [5]:

$$E = \sum_{i=0}^{Np-1} \varphi(m_i, m_i) + 2 \sum_{i=0}^{Np-2} \sum_{j=i+1}^{Np-1} v_i v_j \varphi(m_i, m_j) \tag{6}$$

In the ACELP coder, algebraic codebook is divided into T tracks and each track is consisted of N_P/T nonzero pulses. Thus, the typical excitation vector is denoted by N_P nonzero pulses with amplitude +1 and -1. The search for optimal solution can be performed in N_P/T nested loops by selecting a pulse position from each track. During the search, the contribution of the pulse in ith track in each loop is algebraically added together. Table 1 shows the track format for AMR-WB speech coding algorithm in 12.65 kbps mode [5].

Table 1. Track format for AMR-WB in 12.65 kbps mode [5]

Track	Pulses	Positions in codebook
Track$_0$	i_0, i_4	4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60
Track$_1$	i_1, i_5	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
Track$_2$	i_2, i_6	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
Track$_3$	i_3, i_7	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

The codebook search takes up about one-fourth of overall computational load [40]. So, finding the methods which decrease this computation load and improve the search performance is desirable. In all of the studies on the codebook search methods presented so far, d, ϕ and h are the main components and all suggestions in the field of codebook search are based on some modifications on these components. For example in [40], to decrease the computational load, the approximated correlation matrix (ACM) method has been proposed. In ACM method, some components of autocorrelation matrix, ϕ, which are very small are ignored to reduce the complexity.

Generally, the codebook search complexity is divided into two parts. One part of this complexity is related to computation of d(n) and φ. Another part of this complexity is due to the search loop block. The first part takes up only about one-fourth of codebook search complexity. In most studies, only the first part has been considered. The reduction of computational complexity in this part is performed by omitting the less important components. In this paper, with the aim of more complexity reduction, a new scheme is proposed which affects both the mentioned parts using intelligent optimization algorithms, so it can minimize the computational load considerably.

3. Review of Optimization Algorithms

3.1. GA

The genetic algorithm is a method for solving optimization problems based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. If the chromosome has N_{var} variables (an N-dimensional optimization problem) given by $p_1, p_2, \ldots, p_{Nvar}$, then the chromosome is written as an array with N_{var} elements (Chromosome=$[p_1, p_2, p_3, \ldots, p_{Nvar}]$).

In any optimization problem there is a function as cost function to be minimized. Every element of chromosome must be evaluated by using the cost function. In GA, an initial population of N_{pop} chromosome is defined at first. Considering the dimension of the optimization problem, the size of initial population matrix is $N_{pop}\times N_{var}$ with random-value components:

$$Pop=rand(N_{pop},N_{var}) \tag{7}$$

In the next step, it should be decided on which chromosomes in the initial population have enough fit to survive and possibly reproduce offspring in the next generation. The process of natural selection must occur in each iteration to allow the population of chromosomes to evolve over the generations.

For N_{pop} chromosomes in a given generation, only the top N_{keep} are kept for mating and the rest are discarded to make room for the new offspring. To reproduce the chromosomes in the next generation, crossover and mutation are performed which are GA operations on the remaining individuals of the current generation.

Crossover: Crossover is a genetic operator that combines (mates) two chromosomes (parents) to produce a new chromosome (offspring). The idea behind crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover occurs during evolution according to a user-definable crossover probability. A crossover operator that linearly combines two parent chromosome vectors to produce two new offsprings is as follows:

$$Offspring_1 = a\times Parent_1 + (1-a)\times Parent_2 \tag{8}$$

$$Offspring_2 = (1-a)\times Parent_1 + a\times Parent_2 \tag{9}$$

where a is a random weighting factor (chosen before each crossover operation).

Mutation: As mentioned above, the genetic algorithm uses the chromosomes in the current generation to create the children that make up the next generation. Besides elite children, which correspond to the chromosomes in the current generation with the best fitness values, the algorithm creates (crossover children by selecting vector entries, or genes, from a pair of chromosomes in the current generation and combines them to form a child) and mutation children by applying random changes to a single chromosome in the current generation to create a child:

$$\text{New\_chromosome}_i = Current\_chromosome_i + rand(-1,1)\times (Max\_var - Min\_var) \tag{10}$$

where *Current_chromosome*$_i$ is ith chromosome in the current generation. The *rand(-1,1)* generates the random value between -1 and 1. *Max_var* and *Min_var* are also the maximum and minim range of optimized variable, respectively.

3.2. PSO

In PSO, a number of simple entities, namely the particles, are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best (best-fitness) locations with those of one or more members of the swarm, with some random perturbations.

The algorithm searches a space by adjusting the trajectories of particles as they are conceptualized as moving points in multidimensional space. The individual particles are drawn stochastically toward the positions of their own previous best performance and the best previous performance of their neighbors. If the search space is considered as n-dimensional space, the position and velocity of a particle can be shown by n-dimensional vectors. Consider $x_i^j[t]$ and $v_i^j[t]$ to be the jth element of position and velocity of the ith particle in tth iteration, respectively. The position and velocity of ith particle in $(t+1)$th iteration are defined as [33]:

$$x_i^j[t+1] = x_i^j[t] + v_i^j[t] \tag{11}$$

$$\begin{aligned} v_i^j[t+1] &= wv_i^j[t] + r_1c_1\left(x_{i,best}^j[t] - x_i^j[t]\right) + \\ &\quad r_2c_2\left(x_{gbest}^j[t] - x_i^j[t]\right) \end{aligned} \tag{12}$$

where w is the inertia coefficient and can be constant, variable or random. This coefficient guarantees that the particles which give the best response are not halted and continue their pervious trajectories. The constants c_1 and c_2 are learning coefficients and they are selected in the interval $[0,4]$ and usually $c_1+c_2=4$ [34]. r_1 and r_2 are random numbers with uniform distribution in the interval $[0,1]$. $x_{i,best}[t]$ is the best response that is found by the ith particle until tth iteration and $x_{gbest}[t]$ is the best response of total population until tth iteration.

3.3. BCA

The BCA is an optimization algorithm inspired by the natural foraging behavior of honey bees. The BCA employs a population of different types of bees to find the schedule with minimum *makespan*. The type of a bee is defined based on the behavior she uses to find the food sources. A bee waiting on the dance area for making decision to choose a food source is called onlooker bee; the bee which goes to the food source already visited by herself just before is named as employed bee, and the bee which flies spontaneously in the search space is called scout bee. The algorithm starts with the n scout bees being placed randomly in the search space. Each bee represents a position in the search space. If the project has n activities, the bees will fly in the search space with n dimensions. A position is a candidate for a priority list where each of its elements fixedly represents an activity and its corresponding value shows the priority of that activity.

Initialization is performed by setting the following parameters: Population size, number of scouts (*Scouts*), *Max_Trial*, and Project (*Prj*). *Max_Trial* is the parameter used to identify the food sources that should be abandoned. At initialization step, the number of food sources (*FoodNumber*) will be set to half of Population size, and the population is equally subdivided as employed bees and onlookers. Next food sources will be initialized randomly. *Trial* is the parameter used to be incremented when a food source is not optimized in two consecutive cycles, and *Prj* is the project to be scheduled. At the beginning of each cycle, all the food sources should be evaluated. To evaluate the fitness of a food source, it is needed to generate the schedule from the priority list. Hence, we need to use a schedule generation scheme (SGS). We use serial-SGS that is an activity oriented scheme that generates a schedule in n stages from the priority list [35]. Serial SGS uses two disjoint activity sets at each stage $s \in \{1,2,..., n\}$: the set of scheduled activities and the set Es of eligible activities (i.e. all activities for which all predecessors are scheduled). In each stage, serial-SGS selects one eligible activity $j \in Es$ and schedules it at the earliest precedence and resources feasible time. Next, the set of eligible activities and the resource profiles of partial schedule are updated.

After all the bees are evaluated, each employed bee i selects another employed bee as its own neighbor. After that, a parameter $d \in \{1,2,...,n\}$ will be selected randomly. Each food source will be optimized through following equation:

$$v_{id} = x_{id} + \omega_1 r_{id} (x_{id} - x_{kd}) \tag{13}$$

where i represents the food source which is going to be optimized, $k \in \{1,2,...,FoodNumber\}$ and $d \in \{1,2,...,n\}$ is a randomly chosen index. Parameter ω_1 controls the production of neighbor food sources around x_{id} and represents the comparison of two food positions visually by a bee. Although k is determined randomly, it has to be different from i. The random number r_{id} is selected in range of [-1, 1]. As can be seen, as the difference between the parameters of the x_{id} and x_{kd} is decreased, the perturbation on the position x_{id} is decreased, too. Thus, as the search process approaches the optimum solution in the search space, the step length is adaptively reduced. If a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value. After the employed bees explore the new areas of the food sources, they come into the hive and share the nectar information of the sources with the onlooker bees waiting on the dance area. Sharing the information in the hive, an onlooker bee should employ a decision-making process to select one of the food sources advertised by the employed bees. For this purpose, the probability for each food source k advertised by the corresponding employed bee will be calculated as follows:

$$p_k = \frac{fit(\vec{x}_k)}{\sum_{m=1}^{FoodNumber} fit(\vec{x}_m)} \tag{14}$$

where the probability of proposing the food source by employed bee k, $fit(\vec{x}_k)$, is proportional to the quality of the food source. The quality depends on the *makespan* of the schedule proposed by the food source and is given by:

$$fit(\vec{x}_m) = \frac{1}{makespan_m} \tag{15}$$

where $makespan_m$ is the value of the *makespan* proposed by the food source m. After calculating the probabilities, each onlooker bee employs the roulette wheel to choose a food source advertised by the employed bee k based on its probability. By selecting a food source, the onlooker bee updates its position using the following equation, if the newly discovered food source proposes a schedule with smaller *makespan* than the old one:

$$v_{id} = x_{id} + \omega_2 r_{id} (x_{id} - x_{kd}) \tag{16}$$

where parameter ω_2 controls the importance of the social knowledge provided by the employed bees.

Under this probabilistic approach, the food sources with better schedules attract more onlooker bees. At each cycle of the algorithm, the positions are evaluated and if a food source cannot be improved after a predetermined number of iterations (called *Max_Trial*), then the corresponding food source is abandoned. The *Max_Trial* parameter is determined manually. The abandoned food source is replaced with the new one founded by the scouts. A scout produces a new position randomly and replaces the abandoned food source if the new food source has better nectar.

After each candidate source position v_{ij} is produced and evaluated by the artificial bee, its performance is compared with that of its old one. If the new food source proposes a schedule with smaller *makespan* than the old one, it is replaced with the new one in the memory. Otherwise, the old one is retained in the memory. Finally, by termination of the BCA, the schedule with minimum *makespan* obtained by the population is returned as the output.

3.4. DE

DE has similarities with both GA and PSO. DE uses information of all individuals, and differences between them, to create new solutions for optimization problem. The new solutions are created using difference and trial vectors. To create a new solution y, an old solution a is perturbed using the following rule:

$$y = a + F \otimes (b - c) \tag{17}$$

where b and c are two individuals, randomly selected from population, and $a \neq b \neq c$. Vector F is the scaling factor, and its elements are uniformly distributed random numbers in the range of $[F_{\min}, F_{\max}]$. Operator $\otimes$ is the element-wise multiplication operator.

To create final solution z, crossover operator is applied to y and another randomly selected individual x. There are various methods of crossover. Simplest case is formulated as follows [37]:

$$z_i = \begin{cases} y_i; & r \le CR \text{ or } i = i_0 \\ x_i; & \text{otherwise} \end{cases} \tag{18}$$

where x_i indicates ith element of vector x, scalar r is a uniformly distributed random number in $[0,1]$, CR is the Crossover Rate parameter, and i_0 is a random integer index in the set $\{1,2,3,...,n\}$. It is assumed that the number of search space dimensions (also the number of elements of solution vectors) is equal to n.

In this way, DE uses the information of current population, to create individuals of the next iteration (generation). This process is carried out, until termination conditions are satisfied.

3.5. ICA

The general policy of imperialist competition is used as the basis of the ICA. The process of this algorithm can be summarized as the following seven steps [39]:

Step 1: Select some random points on the function and initialize the empires. As we know the goal of optimization algorithm is to find an optimal solution for a typical problem. The initial population in this algorithm is considered as follows:

$$Country=[p_1, p_2, p_3, \ldots, p_{Nvar}] \tag{19}$$

where $_{Nvar}$ is an N-dimensional optimization problem. It should be mentioned that the variables are as floating point numbers. In any optimization problem, a cost function is considered to be minimized. Every element of country must be evaluated by using the cost function. In ICA, an initial population of N_{pop} country is defined at first. The most powerful countries are selected to form empires and remaining of N_{pop} will be the colonies of corresponding empire N_{col}. Indeed there are two types of countries: imperialist and colony.

Step 2: Move the colonies toward their relevant imperialist (assimilating). In this step, all of the colonies move toward the imperialist. This movement is shown in Fig. 1 in which the colony moves toward the imperialist by x unit. The direction of the movement is the vector from colony to imperialist. Also x is a random variable with uniform distribution, x~$U(0,\beta\times d)$, where β is a number greater than 1 and d is the distance between colony and imperialist. To search different points around the imperialist, a random amount of deviation is defined as θ with uniform distribution, θ~$U(-\gamma,\gamma)$, where γ is a parameter that adjusts the deviation from the original direction.

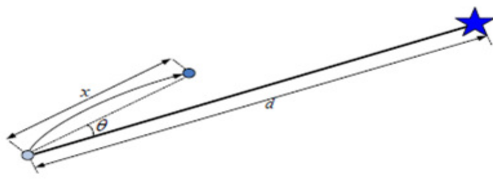


Figure 1. Motion of colonies toward their relevant imperialist in ICA.

Step 3: If there is a colony in an empire which has lower cost than that of imperialist, exchange the positions of that colony and the imperialist.

Step 4: Compute the total cost of all empires (related to the power of both imperialist and its colonies).

$$\begin{aligned} TC_n &= Cost(imperialist_n) + \\ &\xi \times mean\{(Cost(colonies\_of\_empire_n)\} \end{aligned} \tag{20}$$

where ξ is set to -0.1 in our simulations.

Step 5: Pick the weakest colony (colonies) from the weakest empire and give it (them) to the empire that has the most likelihood to possess it (imperialistic competition).

Step 6: Eliminate the powerless empires.

Step 7: If there is just one empire, stop, if not go to Step 2.

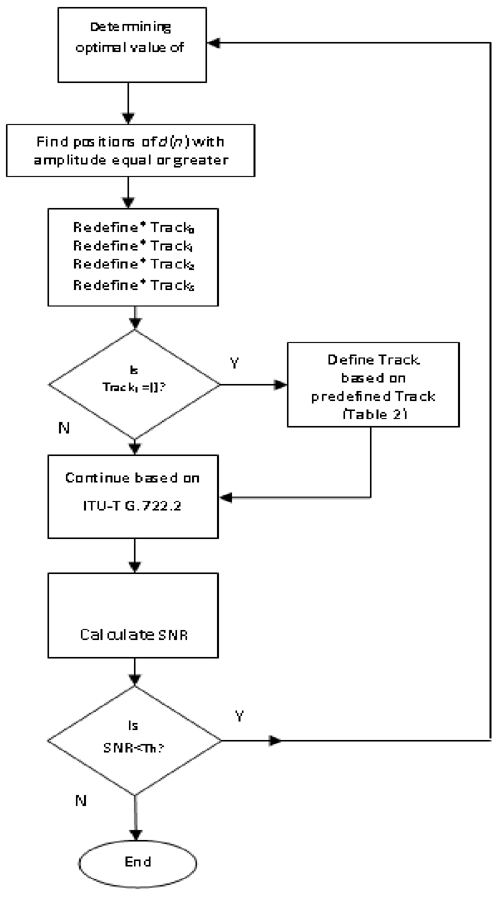
4. Proposed Codebook Search Method

In the following, a method based on the magnitude behavior of inverse filtered target signal (MBIFTS) is presented for codebook search. In other words, the main idea in the proposed method is to focus on *d(n)* behavior. Before the search procedure, the m_1 larger pulses are picked up in *d(n)* sequence. Then, codebook search is only performed for the position of selected pulses, because the maximum of *d(n)* is related to maximum of excitation pluses. The flowchart of proposed algorithm is shown in Fig. 2. As shown in Fig. 2, the predefined tracks are introduced in Table 2.

The advantage of this method is its prevention from unnecessary search processes. To find the mentioned m_1 pulses, let's define the following relation:

$$\beta = \frac{1}{\alpha}\left(\min[|d(n)|]\right); n = 0, \ldots, L \tag{21}$$

where $\alpha \in (0,1]$.



Determining optimal value of
Find positions of d(n) with amplitude equal or greater
Redefine* Track0
Redefine* Track1
Redefine* Track2
Redefine* Track3
Is Track1 =[]?
Y
N
Define Track based on predefined Track (Table 2)
Continue based on ITU-T G.722.2
Calculate SNR
Is SNR<Th?
Y
N
End

* Redefining tracks is based on m_1 positions selected in the pervious step.

Figure 2. Flowchart of the proposed algorithm.

Table 2. Predefined tracks in the proposed algorithm

Track	Pulses	Positions in codebook
Track$_0$	i_0, i_4	4, 8, 12, 16
Track$_1$	i_1, i_5	1, 5, 9, 13,
Track$_2$	i_2, i_6	2, 6, 10, 14
Track$_3$	i_3, i_7	3, 7, 11, 15

The aim of defining this equation is to find the positions of *d(n)* which their amplitudes are larger than β (a factor of minimum *d(n)*). To determine optimum α, using each of the mentioned optimization algorithms in Section 3, a cost function is defined as follows:

$$F = \varepsilon_1 (\frac{SNR}{30}) + \varepsilon_2 \frac{m_1}{L} \qquad (22)$$

This considers simultaneously both the speech quality and complexity reduction criteria. ε_1 is a constant in the range of [0 1] and ε_2 is equal to (1-ε_1). The codebook search is only performed for the position of m_l selected pulses, because the maximum of *d(n)* is related to the maximum of excitation pluses. *L* is equal to 64 (total positions).

The determination of α is a trade-off between computational load reduction and speech quality. Selecting small values for α, results in computational complexity reduction and also quality degradation. On the other hand, selecting large values for α results in less complexity reduction and also higher quality (e.g., in terms of signal to noise ratio (SNR)). For example, if α = 1, then there is no complexity reduction and the proposed algorithm is simplified to the traditional algorithm (recommended in ITU-T G.722.2). It should be noted that since m_l positions are selected, so only the components of ϕ containing these positions are computed.

The best way to define a suitable threshold, introduced in Fig. 2, is to investigate the relation of SNR and the mean opinion score (MOS) using a listening test to find the boundary of SNR in which the quality of speech is not acceptable [40]. It must be noted that MOS value is normally obtained as an average opinion of quality based on asking people to grade the quality of speech signals on a five point scale (Excellent, Good, Fair, Poor, and Bad) under controlled conditions. Also, the International Telecommunication Union (ITU) proposed Recommendation P.862, also known as the perceptual evaluation of speech quality (PESQ) metric [41].

The effect of choosing different values for m_l on the quality of speech in terms of SNR, MOS and PESQ is shown in Figs. 3, 4 and 5, respectively. As can be seen, there is a saturating behavior and increasing m_l (for m_l>32) does not result in noticeable improvement in speech quality. The threshold value is set to 13.2 dB in our simulations. In other words, if the proposed value of α by each of the mentioned optimization algorithms leads to SNR<13.2 dB, then the value is considered invalid. In this way, the effect of choosing different values for α on the number of selected pulses (m_l) is shown in Figure 6.

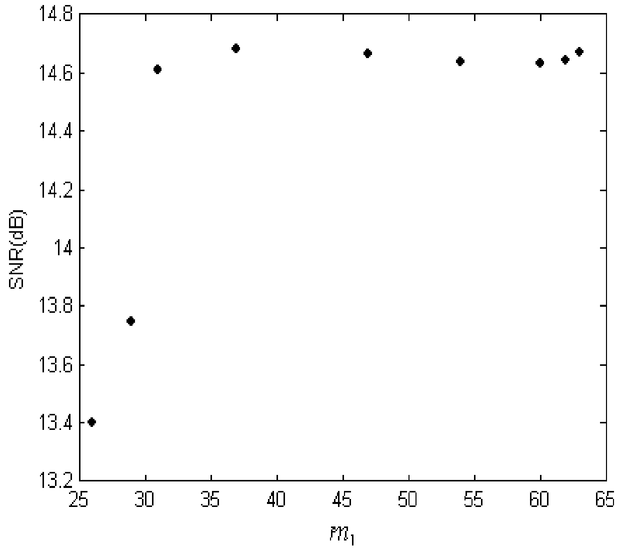


Figure 3. Relation between number of selected pulses (m_l) and speech quality in terms of SNR.

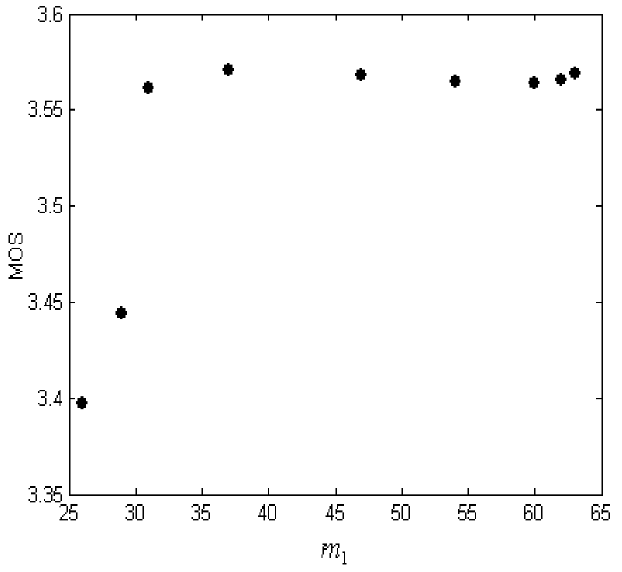


Figure 4. Relation between number of selected pulses (m_l) and speech quality in terms of MOS.

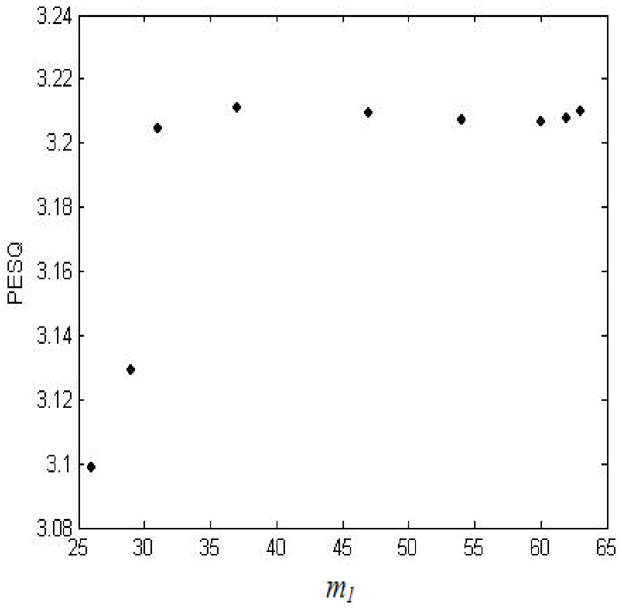


Figure 5. Relation between number of selected pulses (m_l) and speech quality in terms of PESQ.

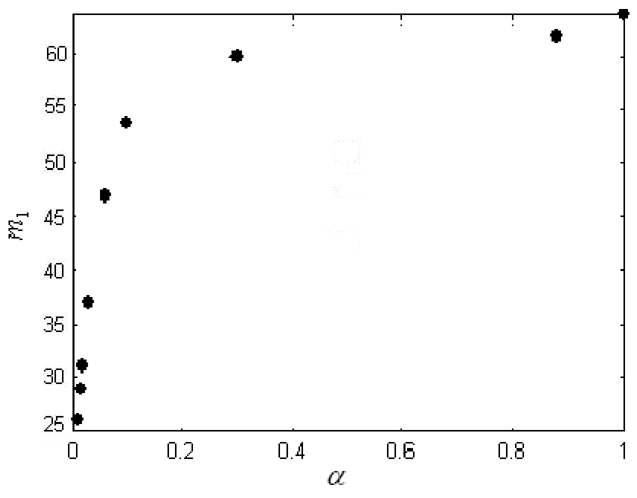


Figure 6. Relation between optimization parameter (α) and number of selected pulses (m_1).

5. Simulation and Experimental Results

In this work, AMR-WB coder in 12.65 kbps mode is implemented. The simulation of encoder and decoder are performed by using MATLAB software and run on PC with Intel Pentium E5300 CPU and 2GB RAM.

The database consists of 1,048,576 vectors from 51 different speakers (twenty five men and twenty six women). This speech database is formed as a part of FARSDAT corpus. FARSDAT is a continuous speech corpus in Farsi language including 6,000 utterances from 300 speakers with various accents [42].

It is noted that α is determined using each of the GA, PSO, BCA, DE and ICA optimization algorithms in this study (Fig. 2). The parameter setting for each of the mentioned algorithms is reported in Table 3 to Table 7, respectively.

Table 3. GA parameters setting

Parameter	Value
Size of population	50
Crossover probability	0.80
Mutation probability	0.3
Number of crossover children	40
Number of mutation children	15
Max_var	1
Min_var	0
Maximum number of iterations	50

Table 4. PSO algorithm parameters setting

Parameter	Value
Size of population	20
Maximum particle velocity	4
c_1	2
c_2	2
Initial inertia weight	0.9
Final inertia weight	0.2
Maximum number of iterations	50

Table 5. BCA parameters setting

Parameter	Value
Size of population	100
Number of scout bees	15
FoodNumber	50
r_{id}	0.5
ω_1	0.8
ω_2	1.2
Maximum number of iterations	50

Table 6. DE algorithm parameters setting

Parameter	Value
Size of population	50
Crossover rate	0.25
F_{min}	0.5
F_{max}	1.5
Maximum number of iterations	50

Table 7. ICA parameters setting

Parameter	Value
Number of countries	40
Number of imperials	10
B	2
γ(rad)	$\pi/4$
ξ	-0.1
Maximum number of iterations	50

The performance of proposed codebook search algorithm, which is equipped with each of the optimization algorithms, is compared to original ITU-T G.722.2 algorithm (Table 8). As can be seen, all of the optimized proposed codebook search methods can obtain the optimal value of α and offer approximately the same quality and reduction in codebook search operations. However, the proposed method that is equipped with BCA performs better in terms of execution time of optimization algorithm while offering noticeable reduction in codebook search operations similar to the other proposed methods.

Table 8. Performance comparison of proposed algorithms with ITU-T G.722.2

Codebook search method	SNR(dB)	Percentage of reduction in codebook search operations as compared to G.722.2	Optimized α	Execution time of optimization algorithm (sec)
Recommended in G.722.2	14.47	NA*	NA*	NA*
MBIFTS+GA	13.25	58.9	0.0104	14,410
MBIFTS+PSO	13.41	59.4	0.0099	12,580
MBIFTS+BCA	13.24	59.1	0.0101	3,780
MBIFTS+DE	13.25	58.9	0.0104	14,940
MBIFTS+ICA	13.25	58.9	0.0104	5,400

6. Conclusion

In this paper, a search codebook method has been proposed for ACELP coder that is based on the magnitude behavior of inverse filtered target signal. Five optimization algorithms (GA, PSO, BCA, DE and ICA) have been used to determine optimum parameter value in the proposed scheme. Experimental results have shown that BCA-optimized codebook search scheme performs better in terms of execution time of optimization algorithm and reduction in the number of operations without significant degradation in quality metrics when employed in an AMR-WB speech coder.

References

[1] J. Adoul, P. Mabilleau, M. Delprat, and S. Morisette, "Fast CELP coding based on algebraic codes," in Proc. Int. Conf. Acoustics, Speech and Signal Processing, pp. 1957-1960, 1987.

[2] ITU-T G.723 Recommendation, Dual-Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kb/s, 1996.

[3] ETSI EN 301 704, Adaptive Multi-Rate (AMR) Speech Transcoding. Digital Cellular Telecommunications System (Phase 2+), version 7.2.1, 1998.

[4] B. Bessette, R. Salami, R. Lefebvre, M. Jelinek, J. Rotola-Pukkila, J. Vainio, H. Mikkola, and K. Jarvinen, "The adaptive multirate wideband speech codec (AMR-WB)," IEEE Trans. Speech and Audio Processing, vol. 10, no. 8, pp. 620-636, 2002.

[5] ITU-T G.722.2 Recommendation, Wideband Coding of Speech at Around 16kbit/s Using Adaptive Multi-Rate Wideband (AMR-WB), 2002.

[6] 3GPP Organizational Partners, Extended Adaptive Multi-Rate wideband (AMR-WB+) Codec (ARIB STD-T63-26.290 V6.2.0), 2005.

[7] M. Jelinek, and R. Salami, "Wideband speech coding advances in VMR-WB standard," IEEE Trans. Audio, Speech and Language Processing, vol. 15, no. 4, pp. 1167-1179, 2007.

[8] C. Laflamme, J-P. Adoul, R. Salami, S. Morissette, and P. Mabilleau, "16 kbps wideband speech coding technique based on algebraic CELP," in Proc. Int. Conf. Acoustics, Speech and Signal Processing, pp. 13-16, 1991.

[9] N. K. Ha, "A Fast method of algebraic codebook search by reordering search sequence," in Proc. Int. Conf. Acoustics, Speech and Signal Processing, vol. 1, pp. 21-24, 1999.

[10] M. A. Ramirez, and M. Gerken, "Joint position and amplitude search of algebraic multipulses," IEEE Trans. Speech and Audio Processing, vol. 8, no. 5, pp. 633-637, 2000.

[11] F. K. Chen, and J. F. Yang, "Maximum take precedence ACELP: a low complexity search method," in Proc. Int. Conf. Acoustics, Speech and Signal Processing, vol. 2, pp. 693-696, 2001.

[12] M. L. Wang, and J. F. Yang, "A generalized candidate scheme of stochastic codebook search for scalable CELP coders," IEE Proceedings-Vision, Image and Signal Processing, vol. 151, no. 5, pp. 443-452, 2004.

[13] R. Pravin Kumar, "High computational performance in code excited linear prediction speech model using faster codebook search techniques," in Proc. IEEE Int. Conf. Computing: Theory and Applications, pp. 458-462, 2007.

[14] M. Halimi, A. Kaddai, and M. Bengherabi, "A new multistage search of algebraic CELP codebooks based on Trellis coding," IEICE Trans. Information Systems, E86-D, pp. 406-411, 2003.

[15] M. Sheikhan, M. Tebyani, and M. Lotfizad, "Continuous speech recognition and syntactic processing in Iranian Farsi language," Int. J. Speech Technology, vol. 1, no. 3, pp. 135-141, 1997.

[16] M. Sheikhan, "Suboptimum extracted features and classifier for speaker-independent Farsi digit recognizer," in Proc. Int. Symp. Telecommunications, pp. 246-249, 2003.

[17] D. Gharavian, M. Sheikhan, and F. Ashoftedel, "Using neutralized formant frequencies to improve emotional speech recognition," IEICE Electronics Express, vol. 8, no. 14, pp. 1155-1160, 2011.

[18] M. Sheikhan, D. Gharavian, and F. Ashoftedel, "Using DTW-neural based MFCC warping to improve emotional speech recognition," Neural Computing and Applications (Published Online 14 May 2011 (doi: 10.1007/s00521-011-0620-8)), 2011.

[19] M. Sheikhan, "Prosody generation in Farsi language," in Proc. Int. Symp. Telecommunications, pp. 250-253, 2003.

[20] M. Sheikhan, M. Nasirzadeh, and A. Daftarian, "Text to speech for Iranian dialect of Farsi language," in Proc. Workshop Farsi Computer Speech, pp. 39-53, 2006.

[21] M. Sheikhan, V. Tabataba Vakili, and S. Garoucy, "Complexity reduction of LD-CELP speech coding in prediction of gain using neural networks," World Applied Sciences Journal, vol. 7 (Special Issue of Computer & IT), pp. 38-44, 2009.

[22] M. Sheikhan, V. Tabataba Vakili, and S. Garoucy, "Codebook search in LD-CELP speech coding algorithm based on multi-SOM structure," World Applied Sciences Journal, vol. 7 (Special Issue of Computer & IT), pp. 59-68, 2009.

[23] M. Sheikhan, and S. Garoucy, "Reducing the codebook search time in G.728 speech coder using fuzzy ARTMAP neural networks," World Applied Sciences Journal, vol. 8, no. 10, pp. 1260-1266, 2010.

[24] M. Sheikhan, and S. Garoucy, "Hybrid VQ and neural models for ISF quantization in wideband speech coding," World Applied Sciences Journal, vol. 10 (Special Issue of Computer & Electrical Engineering), pp. 59-66, 2010.

[25] M. Sheikhan, D. Gharavian, and A. Eslamzadeh, "Enhancement of LPC-10 speech coder using LSP Parameters and Neural Vector Quantizers," World Applied Sciences Journal, vol. 10 (Special Issue of Computer & Electrical Engineering), pp. 41-48, 2010.

[26] M. Sheikhan, and S. Garoucy, "Computational complexity reduction of AMR-WB speech coding algorithm using new GA-optimized fast codebook search techniques," World Applied Sciences Journal, vol. 14, no. 1, pp. 63-70, 2011.

[27] M. Sheikhan, and S. Garoucy, "Prediction of gain in LD-CELP using hybrid genetic/PSO-neural models," Journal of Advanced Researches in Computer, vol. 1, no. 3, pp. 1-12, 2011.

[28] D. Gharavian, M. Sheikhan, A. R. Nazerieh, and S. Garoucy, "Speech emotion recognition using FCBF feature selection method and GA-optimized fuzzy ARTMAP neural network," Neural Computing and Applications (Published Online 27 May 2011 (doi: 10.1007/s00521-011-0643-1)), 2011.

[29] M. Sheikhan, M. K. Safdarkhani, and D. Gharavian, "Emotion recognition of speech using small-size selected feature set and ANN-based classifiers: a comparative study," World Applied Sciences Journal, vol. 14, no. 4, pp. 616-625, 2011.

[30] M. Sheikhan, M. Tebyani and M. Lotfizad, "Using symbolic and connectionist approaches to automate editing Persian sentences syntactically," in Proc. Int. Conf. Intelligent and Cognitive Systems, pp. 250-253, 1996.

[31] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Learning, Addison Wesley, 1989.

[32] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization," in Proc. IEEE Int. Conf. Neural Networks, vol. 4, pp. 1942-1948, 1995.

[33] Y. Shi, and R. C. Eberhart, "Parameter selection in particle swarm optimization," in Proc. Int. Conf. Evolutionary Programming, pp. 591-600, 1998.

[34] Y. Shi, and R. C. Eberhart, "Empirical study of particle swarm optimization," in Proc. IEEE Int. Conf. Evolutionary Computation, pp. 1945-1950, 1999.

[35] N. Quijano, and K. M. Passino, "Honey bee social foraging algorithms for resource allocation, part I: algorithm and theory," in Proc. American Control Conf., pp. 3383-3388, 2007.

[36] N. Quijano, and K. M. Passino, "Honey bee social foraging algorithms for resource allocation, part II: application," in Proc. American Control Conf., pp. 3389-3394, 2007.

[37] R. Storn, and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces", J. Global Optimization, vol. 11, no. 4, pp. 341-359, 1997.

[38] A. K. Qin, and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in Proc. IEEE Cong. Evolutionary Computation, vol. 2, pp. 1785-1791, 2005.

[39] E. Atashpaz-Gargari, and C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," in Proc. IEEE Cong. Evolutionary Computation, pp. 4661-4667, 2007.

[40] ITU-T P.830 Recommendation, Subjective Performance Assessment of Telephone-Band and Wideband Digital Codecs, 1996.

[41] ITU-T P.862 Recommendation, Perceptual Evaluation of Speech Quality (PESQ), an Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs, 2001.

[42] M. Bijankhan, J. Sheikhzadegan, M. R. Roohani,Y. Smareh, C. Lucas, and M.Tebiani, "The speech database of Farsi spoken language," in Proc. Australian Conf. Speech Science and Technology, pp. 826-831, 1994.