ISSN No. 0976-5697

Volume 3, No. 7, Nov-Dec 2012

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Optimization Techniques for Digital Filter Design: A Review

G. S. Gawande
Department of Electronics & Telecommunication,
SSGMCE, Shegaon,
Maharashtra, India

Rupesh D. Sushir*
Department of Electronics & Telecommunication,
SSGMCE, Shegaon, Maharashtra, India
rupeshsushir@gmail.com

Abstract: The basic problem is to identify the alternative means of achieving a given objective and then to select the alternative that accomplishes the objective in the most efficient manner, subject to constraints on the means which is referred as an optimization. In this paper a brief study of different optimization techniques along with their classification, used for digital filter design and analysis are reviewed along with Integer linear Programming algorithm which provides effective results in constraint with time.

Keywords: Optimization, Simulated Annealing (SA), Particle Swarm Optimization (PSO)

I. INTRODUCTION

Optimization, in the straightforward interpretation, is the process of finding the optimum value of a given function, the objective function, on a particular domain, possibly with a number of additional constraints. An optimum can be either a maximum or a minimum depending on the problem formulation, but since it is straightforward to turn a minimization problem into a maximization problem, and vice versa. The great divide in classification of optimization problems depends on the domain on which the objective function is defined. A general optimization problem is specified by a set of problem instances and each instance can be formalized as a pair (S, f) where S is the domain, or solution space, comprising of all possible solutions and the objective, or cost, function f is a mapping $f : S \to R$ associating every point s 2 S with a real-valued cost [1]. With the above definitions the problem is to find the globally optimal solution s_{opt} , which satisfies

$$f(s_{opt}) \leq f(s), s \in S$$

and the corresponding optimal value of the cost function, $f_{opt} = f(s_{opt})$. The optimization problem is based on the computational time needed to solve a problem as the size grows. The optimization is classified in next section.

A. Classification of continuous optimization problems:

Optimization problems are classified as either convex or non-convex based on the domain and the cost function. The domain is convex if a straight line between any two point's C_1 and C_2 in the domain is also part of the domain and the cost function is convex if its value at any point along the straight line between any two point's C_1 and C_2 in the domain has an upper bound in the chord through (C_1, f (C_1)) and (C_2, f (C_2) as shown in the figure 1.1(a)

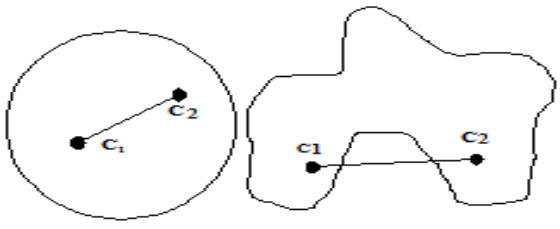

Figure 1.1(a) Examples of a convex (left) and a non-convex domain (right)

A polynomial function of the problem size and a polynomial function which require a super-polynomial can be distinguished which have a solution time, with respect to the best known algorithm, execution time in terms of their size. Due to concept which includes the growth rates such as $a > 1$, n^n and n!. The term exponential is used to describe the growth rate rather than super-polynomial.

B. Optimization algorithms:

An optimization algorithm is closely allied to the particular optimization problem but in general there is a trade-off from adapted algorithms applicable only to a restricted subset of a problem to general algorithms which can handle a large variety of problems without modification. The major consequence of a general algorithm is that it is time consuming where an efficient general algorithm explicitly "fast enough" in most cases. In order to compare the executing time complexity of different algorithms, these algorithms are elaborated below.

C. Simulated Annealing (SA) algorithm:

The simulated annealing belongs to class of Natural Algorithms as it posses property to manage itself into optimal states with respect to its surrounding. This algorithm is based on similarity between the behaviors of a melted solid which is transformed into a perfect crystal when it is slowly cooled (annealed) past its melting point.

In the liquid state as the atoms will move around and randomly rearrange them but due to decrease in surrounding temperature the system is cooled and they are less likely to leave their place in the forming structure. The lowest possible energy of the system can be achieved when the cooling proceeds slowly enough and the crystalline state reached at zero temperature, the atoms fixed in a perfect lattice structure. In order to deal with the behavior of single atom which has no knowledge of energy of the system as whole in spite [1] it only interacts with its local neighbors. In order to give solution to the above mention problem, a state of the forming solid would correspond to a possible solution to the optimization problem and the energy of that state would correspond to the quality, or cost, of that solution. In short, by applying a combinatorial optimization problem to a statistical mechanics framework, it is possible to reach a globally optimal or nearly optimal solution without considering anything but local interactions in the

system. Due to the simplicity and vigorous behavior of the SA algorithm a large amount of work has been done which reports successful application of simulated annealing to such diverse problem areas as digital filter design, VLSI design, molecular biology and landscape management.

D. FIR Filter Design with Simulated Annealing Algorithm:

Boltzmann annealing (Ingber, 1989), which is defined by the following elements:

$$g(b_n) = \left[(2\pi t)^{\frac{-(N-1)}{2}} \right] e^{\frac{\Delta b^2 n}{2T}}$$

Where g (b_n) is the probability density function (pdf) of the filter coefficients b_n deviation, $\Delta b_n = b_n^{(t+1)} - b_n^t$ is the deviation from state (filter) *i* to *i* +1, and *T* is a measure of the fluctuations of the Boltzmann distribution *g* in the $N-1$ dimension of the filter coefficients. $h(b_n) = \frac{1}{1+e^{\frac{\Delta E}{T}}}$

where h(b_n) is the (PDF) probability for acceptance of new cost-function (energy state) and $\Delta E = E_{(k+1)} - E_{(k)}$ represents the energy difference between the present and the previous values of the cost-function [2] i.e. filter error $T(k) = \frac{T_0}{\ln k}$, where *T* (*k)* is the schedule of annealing the temperature *T* in annealing-time steps.

Simulated annealing is beneficial with arbitrary systems and cost functions for any given problem as well as it statistically guarantees finding an optimal solution for any optimization problem, besides this if annealing with a (1/log k) is very time consuming for complex cost function makes major drawback of SA.

E. Genetic algorithm:

In order to search for an optimal solution to the evolution function of an optimization problem Genetic Algorithm plays vital role. From the various researchers view, GAs is also useful for functional optimization given by De Jong [5] and a detailed mathematical model of a GA was proposed by Goldberg [6].

GA distinguishes from classical optimization and search methods in various aspects. GAs operates on group of trial solutions in parallel instead of focusing on a single solution. Where they manipulate a *population* of individuals in each *generation* (iteration) where each individual, termed as the *chromosome*, represents one candidate solution to the problem. Within the population, fit individuals survive to reproduce and their genetic materials are recombined to produce new individuals as *offspring*. The genetic material is modeled by some finite-length data structures. As in nature, *selection* provides the necessary driving mechanism for better solutions to survive. Each solution is associated with a *fitness* value that rejects how well it is compared with other solutions in the population. The recombination process is simulated through a *crossover* mechanism that exchanges portions of data strings between chromosomes. New genetic material is also introduced through *mutation* that causes random alterations of the strings. The frequency of occurrence of these genetic operations is controlled by certain pre-set probabilities. The selection, crossover, and mutation processes constitute the basic GA cycle or generation, which is repeated until some pre-determined criteria are satisfied. Through this process, successively better and better individuals of the species are generated. In other words with the help of GA, on can find the solution to problem with few efforts. With the help of schematic representation of the genetic search approach shown in figure 2.1 GA can be explained with four fundamental steps:

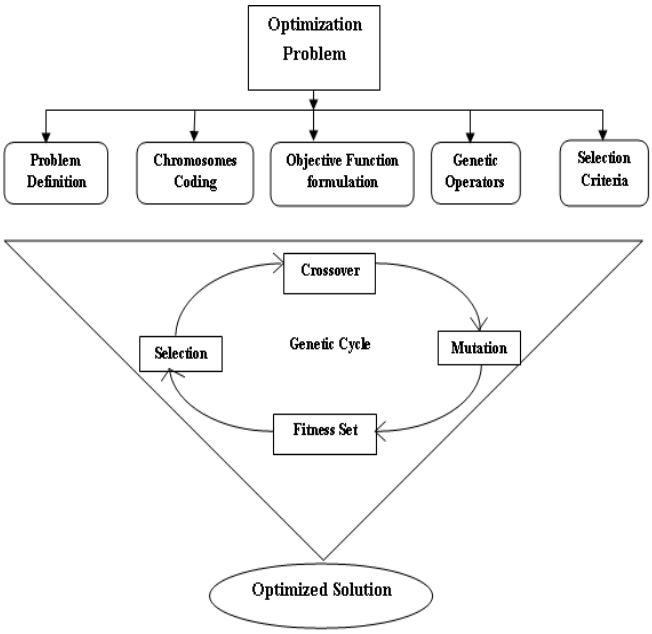

Figure 2.1 Conceptual representation of the optimization process through a genetic algorithm.

Step 1: Generate an initial population of random solutions (chromosomes).

Step 2: According to the fitness criteria, judge the chromosomes and create the best set of chromosomes by selecting a number of chromosomes that satisfy the requirements imposed on the solution.

Step 3: Among the best set satisfies fully the requirements imposed on the solution, output that chromosome as the required solution, and stop. Otherwise, execute *Step 4.*

Step 4: To generate more chromosomes crossover is applied between pairs of chromosomes and certain chromosomes are chosen at random mutations, and repeat process from *Step 2.*

II. FIR FILTER DESIGN WITH GENETIC ALGORITHM

A. Population:

The population of the genetic algorithm consists of a given number of individuals representing each one a possible optimum FIR filter. Each individual chromosome is represented by a set of FIR filter coefficients, initialized setting each coefficient with a real random value of a Gaussian distribution.

B. Fitness Function:

At each new generation of the genetic algorithm, the offspring is created based on the fitness function. As the objective of the optimization is the minimization of equation 3, the fitness is defined as the inverse of the error:

$$f(x) = \{w(\omega)[H_d(e^{j\omega}) - H(e^{j\omega})]\}^{-1}$$

C. Selection:

Based on the fitness function, each individual is selected using a roulette-wheel selection method, meaning that to each individual a slice in the wheel is assigned, its width proportional to the individual fitness. In addition, the algorithm uses elitism in the selection, meaning that a predefined number of best individuals are always selected for the next offspring.

D. Crossover:

The crossover genetic operation is defined for the current implementation as follows. A randomly chosen crossover position in the parents filter coefficients set is defined. Then the parent's filter coefficients are swapped over this randomly generated crossover position.

E. Mutation:

With a predetermined probability, each parent chromosome (filter coefficient set) is mutated adding a small random value of a Gaussian distribution.

Genetic algorithm finds useful application in solving optimization problem described with chromosomes encoding. It also solves problem with multiple solution and due to its independent nature towards error surface, it is useful in solving multi-dimensional, non-differential, non-continuous, and even non-parametrical problems. In addition to above advantages GA possesses some disadvantages like, as only good chromosomes block cross-over, the poorly known fitness functions which creates these blocks. Due to this fact certain optimization problems cannot be solved by means of genetic algorithms. When the populations have lot of subjects, genetic algorithm fails to find global optimal.

F. Particle Swarm Optimization:

In 1995 another optimization technique based on population technique was developed by Dr. Eberhart and Dr. Kennedy [7], [8] which is inspired from social behavior of bird flocking or fish schooling.

PSO is closely related to Genetic Algorithms (GA) [4]. The system is initialized with a population of random solutions and searches for optima by updating generations. PSO is different from GA in the sense that it has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

PSO is classified on the basis of optimal solution into three categories viz, pbest i.e. each particle keeps record of its coordinates in the problem space which are associated with the best solution for it and it stores these best value. PSO finds another best value [4] amongst in the neighbors of the particle and this location is termed as lbest while a particle takes all the population as its topological neighbors, the best value is a global best and is called gbest.

At each time step, PSO changes the velocity of (accelerating) each particle toward its [4] pbest and lbest locations in its basic version. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and lbest locations.

Mathematically, velocities of the particle vectors are modified according to the following equation:

$$V_i^{(k+1)} = w * V_i^k + C_1 * rand_1 * (pbest_i^k - S_i^k) + C_2 * rand_2 * (gbest_i^k - S_i^k) \quad \ldots. \quad (1)$$

Where V_i^k the velocity of i^{th} particle is vector at k^{th} iteration; w is the weighting function; C_1 and C_2 are the positive weighting factors; $rand_1$ and $rand_2$ are the random numbers between 0 and 1; S_i^k is the current position of i^{th} particle vector h(n) at k^{th} iteration; $pbest_i^k$ is the personal best of the i^{th} particle at the k^{th} iteration; $gbest^k$ is the group best of the group at the k^{th} iteration. The searching point in the solution space may be modified by the following equation:

$$S_i^{(k+1)} = S_i^k + V_i^{(k+1)} \quad \ldots\ldots\ldots (2)$$

The first term of equation (1) is the previous velocity of the particle vector. The second and third terms are used to change the velocity of the particle vector. Without the second and third terms, the particle vector will keep on ''flying'' in the same direction until it hits the boundary. Namely, it corresponds to a kind of inertia represented by the inertia constant, *w* and tries to explore new areas.

When PSO is judged in terms of execution time and cost against other methods of optimization, it proves its robustness. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

III. FIR FILTER DESIGN WITH PARTICLE SWARM OPTIMIZATION

A swarm can be described as a population of interacting elements that is able to optimize some global objective through collaborative search of a space. Interactions that are relatively local (topologically) are often emphasized. There is a general stochastic (or chaotic) tendency in a swarm for individuals to move toward a center of mass in the population on critical dimensions, resulting in convergence.

The PSO is applied to the FIR filter design problem [2] with the following set of operations:

A. Loop:

For i=1 **to number of individuals** (b_n's FIR filter coefficients).

If $G(\vec{x_i}) > G(\vec{p_i})$ **then do,** (x_i is the current particle, FIR filter, of the loop)**.**

For d =1 **to dimensions** (The dimension is equals the FIR filter length).

$\boldsymbol{p_{id} = x_{id}}$ **,** ($\boldsymbol{p_i}$ is the $\boldsymbol{x_i}$ particle with best fitness value).

Next d

End do

g = i

For j **= indexes of neighbors**

If $G(\vec{x_i}) > G(\vec{p_i})$ **then** $g = j$, where $G = W(\omega)[H_d(e^{j\omega}) - H(e^{j\omega})]^{-1}$ is the particle fitness function, the inverse of the FIR filter error.

Next *j*
For *d* =1 to number of dimensions

$$v_{id}(t) = v_{id}(t-1) + \varphi_1(p_{id} - x_{id}(t-1)) + \varphi_2(p_{gd} - x_{id}(t-1))$$

where φ_1 , φ_2 are random numbers drawn from a uniform distribution with predefined up and down limits.

$v_{id}(t) \in (-V_{\max}, V_{\max})$ where v_i is the particle velocity term and $V_{\max}$ is a limiting term to prevent explosion in the particle movement through the hyperspace.

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t)$$

Next d
Next *i*

B. Until criterion:

In order to minimize the ripples with hardware constraint in less time an improved optimization algorithm (Integer Linear Programming) which involves minimizing the number of nonzero bits in each coefficient without violating the filter specifications within the pass and stop bands can be implemented**.** This algorithm offers fast computation because of already sorted search space. This approach achieves comparable reductions to ripple because of multiple optimizations iterations.

IV. REFERENCES

[1]. Persson P., "Annealing Based Optimization Methods for Signal Processing Applications", Blekinge Institute of Technology Dissertation Series 2003:01 ISSN 1650-2159.

[2]. Flávio C. A. Teixeira, "Optimum finite impulse response digital filter design using computational intelligence based optimization algorithms", IADIS International Conference Intelligent Systems and Agents 2007, ISBN: 978-972-8924-39-3.

[3]. Sabbir U. Ahmad, "Design of Digital Filters Using Genetic Algorithms", University of Victoria, 2008.

[4]. Sangeeta Mondal et al, "Linear Phase High Pass FIR Filter Design using Improved Particle Swarm Optimization", World Academy of Science, Engineering and Technology 60 2011.

[5]. K. D. Jong, "Hybrid methods using genetic algorithms for global optimization", IEEE Trans. Systems Man Cybernatics, vol. 10, no. 9, pp. 566 {574, September 1980.

[6]. D. E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning. San Francisco: Addison-Wesley, 1989.

[7]. J. Kennedy, R. Eberhart, Particle Swarm Optimization, in Proc. IEEE int. Conf. On Neural Network, 1995.

[8]. R. Eberhart, Y. Shi, Comparison between Genetic Algorithms and Particle Swarm Optimization, Proc. 7th Ann. Conf. on Evolutionary Computation, San Diego, 2000