Structured Stochastic Optimization Strategies for Problems with Ill-conditioned Error Surfaces

Siddharth Pal, D. J. Krusienski, and W. K. Jenkins
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA

Abstract—**This paper compares the performance of several structured optimization strategies in adaptive signal processing problems that are characterized by ill-conditioned error surfaces. The genetic algorithm (GA), the particle swarm optimization (PSO) algorithm, and a new constrained random search (CRS) algorithm [8] are considered. When applied to adaptive filters, these structured stochastic search strategies are independent of the adaptive filter structure and are capable of converging to the global solution when applied in circumstances that create multi-modal mean squared error surfaces.**

I. INTRODUCTION

In many adaptive signal processing problems the mean squared error surface that exists as a hyper-surface in the multidimensional parameter space may be ill-conditioned in the sense that it is a non-quadratic surface that possesses local minima and saddle points, in addition to a global minimum that represents the optimal solution. It is well known that such surfaces inhibit efficient optimization for certain classes of adaptive systems, such as infinite impulse response (IIR) adaptive filters, nonlinear polynomial adaptive filters, and neural networks. One such situation occurs in independent component analysis (ICA), where it is well known that the mutual information function exhibits multi-modality that tends to inhibit the success of gradient descent algorithms.

Stochastic optimization algorithms aim at increasing the probability of encountering the global minimum, without performing an exhaustive search of the entire parameter space. Unlike gradient descent techniques, the performance of stochastic optimization techniques in general is not dependent upon the filter structure. Therefore, these types of algorithms are capable of globally optimizing any class of adaptive filter structures or any types of objective functions.

II. STRUCTURED STOCHASTIC OPTIMIZATION

The foundation of a *structured* stochastic search is to *intelligently* generate and modify the randomized estimates in a manner that efficiently searches the error space, based on some previous or collective information generated by the search. Several different structured stochastic optimization techniques can be found in adaptive filtering literature, most notably simulated annealing [1], evolutionary algorithms such as the genetic algorithm [9], and swarm intelligence algorithms such as particle swarm optimization [5][6][7]. One interesting item to note is that all of the prominent structured stochastic optimization techniques are inspired by a natural or biological process.

A. Evolutionary Algorithms

Evolutionary algorithms (EA) begin with a random set of possible solutions (the unknown parameters to be optimized), termed the population [4]. Each possible solution in the population is termed an individual. Each individual's set of parameters is termed a chromosome or genome, and each parameter is termed a gene. Depending on the nature of the problem, the chromosomes may represented as real numbers or can be encoded as binary strings.

For every generation the fitness of each individual is evaluated by a predetermined fitness function that is assumed to have an extremum at the desired optimal solution. An individual with a fitness value closer to that of the optimal solution is considered better fit than an individual with a fitness value farther from that of the optimal solution. The population is then evolved based on a set of principles rooted in evolutionary theory such as natural selection, survival of the fittest, and mutation. Natural selection is the mating of the fittest individuals (parents) within the population to produce a new individual (offspring). This equates to randomly swapping corresponding parameters (crossover) between the parents to produce a new, potentially fit individual. The new offspring then replace the least fit individuals in the population, which is the survival of the fittest facet of the evolution. A portion of the population is then randomly mutated in order to add new parameters to the search. The expectation is that only the offspring that inherit the best parameters from the parents will survive and the population will continually converge to the best possible fitness that represents the optimal or suitable solution.

Previous work on EAs for adaptive filtering, specifically the GA, shows that the GA is capable of globally optimizing both IIR [9] and nonlinear [10] filter structures. The performance of the GA is examined for general nonlinear recursive adaptive filters in [10], along with a proof of convergence for the estimation error.

B. Particle Swarm Optimization

Particle swarm optimization was first developed in 1995 by Eberhart and Kennedy [2]. Based on the notion of swarm intelligence of insects, birds, etc. the algorithm attempts to mimic the natural process of group communication of individual knowledge that occurs when such swarms flock, migrate, forage, etc. in order to achieve some optimum property such as configuration or location.

The conventional PSO algorithm begins by initializing a random swarm of M particles, each having R unknown parameters to be optimized. At each epoch, the fitness of each particle is evaluated according to the selected fitness function. The algorithm stores and progressively replaces the most fit parameters of each particle (*pbest*$_i$, i=1,2,...,M) as well as a single most fit particle (*gbest*) as better fit parameters are encountered. The parameters of each particle (pi) in the swarm are updated at each epoch (n) according to the following equations:

$$\overline{vel_i}(n) = w * \overline{vel_i}(n-1) + acc_1 * diag[e_1, e_2, ..., e_R]_{i1} * (gbest - p_i(n-1))$$
$$+ acc_2 * diag[e_1, e_2, ..., e_R]_{i2} * (pbest_i - p_i(n-1)) \quad (1)$$
$$p_i(n) = p_i(n-1) + \overline{vel_i}(n) \quad (2)$$

where $\overline{vel_i}(n)$ is the velocity vector of the i^{th} particle, e_r is a vector of random values within in the interval (0,1), acc_1 and acc_2 are the acceleration coefficients toward *gbest* and *pbest*$_i$ respectively, and w is the inertia weight.

Each particle is influenced in a direction determined by the previous velocity and the location of *gbest* and *pbest*$_i$. Each particle's previous position (*pbest*$_i$) and the swarm's overall best position (*gbest*) are meant to represent the notion of individual experience memory and group knowledge of a "leader or queen", respectively, that emerges during the natural swarming process. The acceleration constants are typically chosen in the interval (0,2) and serve dual purposes in the algorithm. For one, they control the relative influence toward *gbest* and *pbest*$_i$, respectively, by scaling each resulting distance vector. Secondly, the two acceleration coefficients combined form what is analogous to the step size of an adaptive algorithm. Acceleration coefficients closer to 0 will produce fine searches of a region, while coefficients closer to 1 will result in lesser exploration and faster convergence. Setting the acceleration greater than 1 allows the particle to possibly over-step *gbest* or *pbest*, resulting in a broader search. The random e_i vectors have R different components, which are randomly chosen from a uniform distribution in the interval (0,1). This allows the particle to take constrained randomly directed steps in a bounded region between *gbest* and *pbest.*

When a new *gbest* is encountered during the update process, all other particles begin to swarm toward the new *gbest*, continuing the directed global search along the way. The search regions continue to decrease as new *pbest*$_i$*'s* are found within the search regions. When all of the particles in the swarm have converged to *gbest*, the *gbest* parameters characterize the minimum error solution obtained by the algorithm. One of the key advantages of PSO is the ease of implementation in both the context of coding and parameter selection.

C. A New Constrained Random Search Algorithm

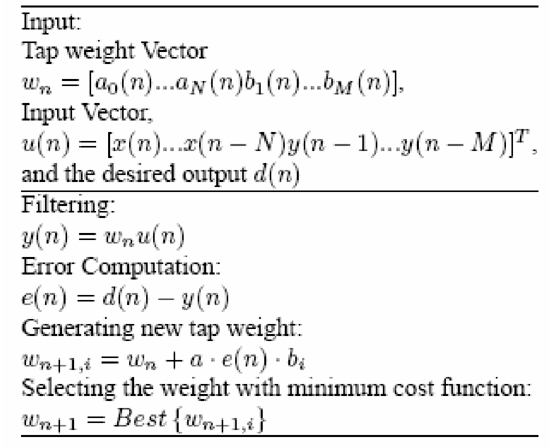
The form of the new constrained random search (CRS) method proposed here was motivated by a study of simulated annealing [1]. The SA algorithm starts with defining an initial configuration C_0 and an initial temperature $T=T_0$. It then generates a sequence of configurations $N=N_0$. The temperature is then reduced and a new number of steps to be performed at that temperature are determined. A candidate configuration is accepted if its cost is less than that of current configuration. If the cost of the new configuration is larger than the cost of the previous configuration it can still be accepted with a certain probability. Typically, SA is not a population based search method, so instead of taking N_0 steps one after another, we decided to instead initialize N_0 configurations around a configuration (filter coefficient vector) and then evaluate the fitness of all these. The fittest among the population is selected and the process is repeated.

The weight initialization equation for the CRS algorithm proposed here is given by:

$$\bar{w}_i(n+1) = \bar{w}(n) + a(n) * e(n) * \bar{b}_i(n) \quad (3)$$

where $\bar{w}_i(n+1)$ is a newly generated i^{th} configuration (weight vector), $\bar{w}(n)$ is the previous best configuration, *a(n)* is a parameter which decreases as the number of iteration increases, *e(n)* is the instantaneous error value of the previous configuration and $\bar{b}_i(n)$ is a random vector consisting of uniformly distributed random numbers between [-1,1]. It was discovered experimentally that the inclusion of the instantaneous error term *e(n)* in equation (3) gives a faster convergence when $\bar{b}_i(n)$ training IIR adaptive filters.

The product of *e(n)* and $\bar{b}_i(n)$ can be viewed as a noisy gradient. So basically we start with a random filter weight vector and then generate $N=N_0$ new weight vectors around it, governed by equation (3). The starting value of *a(n)* is kept high so that the algorithm searches the space to find the global valley and slowly it decreases to fine tune the solution. The following table summarizes the CRS algorithm.



Input:
Tap weight Vector
$w_n = [a_0(n)...a_N(n)b_1(n)...b_M(n)]$,
Input Vector,
$u(n) = [x(n)...x(n-N)y(n-1)...y(n-M)]^T$,
and the desired output $d(n)$

Filtering:
$y(n) = w_n u(n)$
Error Computation:
$e(n) = d(n) - y(n)$
Generating new tap weight:
$w_{n+1,i} = w_n + a \cdot e(n) \cdot b_i$
Selecting the weight with minimum cost function:
$w_{n+1} = Best\{w_{n+1,i}\}$

Table 1. Summary of CRS algorithm

III. EXPERIMENTAL EXAMPLES

In this section two experimental examples are presented to demonstrate the performance of structured stochastic optimization algorithms on system identification problems that have multi-modal mean-squared-error (mse) surfaces. The learning characteristics shown below were produced by ensemble averaging over fifty independent trials and time averaging over a window of size 100.

Example 1. (*IIR system, matched order, colored noise input, SNR=20dB)*: This example is taken from [3]. The adaptive system is excited by a colored noise input obtained by filtering white noise. The transfer functions of the plant Hp(z), the adaptive filter Hf (z), and the coloring filter Hc(z) are given by:

$$H_p(z^{-1}) = \frac{1}{(1-0.7z^{-1})^2} \tag{4}$$

$$H_f(n,z^{-1}) = \frac{b_0(n)}{1+a_1(n)z^{-1}+a_2(b)z^{-2}} \tag{5}$$

$$H_c(z^{-1}) = (1-0.7z^{-1})^2(1+0.7z^{-1})^2 \tag{6}$$

It has been shown previously that two minima exist on the mse surface for this example [3]. The learning characteristics for a population size of 25 are shown in figure 1 for standard particle swarm optimization (PSO), for modified particle swarm optimization (MPSO) [6] (that includes mutation and re-randomization about *gbest* each time a new gbest is found) and the genetic algorithm (GA). Figure 2 shows the same experiment using the CRS algorithm. It can be seen from figure 1 that the standard PSO algorithm converges quickly, but tends to "stagnate" considerably above the –20 dB noise floor. Both the MPSO algorithm and the GA converge to the –20 dB noise floor, although the MPSO algorithm reaches the noise floor faster. From the results shown in figure 2 it appears that the CRS algorithm reaches the noise floor faster than the algorithms shown in figure 1, although the improved rate of convergence is marginal.

Example 2. (*IIR system, reduced order, colored noise input:* For this example the plant is given by:

$$H_p(z^{-1}) = \frac{1}{(1-0.6z^{-1})^3} \tag{7}$$

$$H_f(n,z^{-1}) = \frac{b_0(n)}{1+a_1(n)z^{-1}+a_2(b)z^{-2}} \tag{8}$$

$$H_c(z^{-1}) = (1-0.6z^{-1})^2(1+0.6z^{-1})^2 \tag{9}$$

The adaptive system is excited by a colored input generated by filtering white noise with the coloring filter given by equation (9). The coloring, in combination with the reduced order, creates the bimodal error surface shown in figure 3 [3]. The experimental results for this example using a population of 25 and 50 are shown in figures 4 and 5, respectively. As in the previous example, the PSO algorithm tends to stagnate at a level above the noise floor, whereas the MPSO and the GA algorithms are both able to reach the global minimum. In this case the global minimum, approximately –17.0 dB, is determined by the mismatch between the second order adaptive system and the third order plant. In this case too the CRS algorithm performs somewhat better than the other algorithms with a comparable population size.

IV. CONCLUSIONS

The population-based methods discussed here appear to be able to locate global minima. In general on nonconvex error surfaces population based methods out perform gradient based algorithms [6][7][8]. In both examples the CRS algorithm performs better than the PSO algorithm, although it is not yet known if this is a general result.

REFERENCES

[1] Aarts, E., Korst, J., *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, 1989.

[2] Eberhart, R. C. and Kennedy, J. "A new optimizer using particle swarm theory," Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.

[3] H. Fan and W. K. Jenkins, "A new adaptive IIR filter, "*IEEE Transactions on Circuits and Systems*, vol. CAS-33, pp. 939–947, 1986.

[4] Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[5] Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm intelligence,* San Francisco: Morgan Kaufmann Publishers, 2001.

[6] Krusienski, D.J. and Jenkins, W.K., "Particle Swarm Optimization for Adaptive IIR Filter Structures," *Proc. of the 2004 Congress on Evolutionary Computation*, June 2004.

[7] Krusienski, D.J. and Jenkins, W.K., "The Application of Particle Swarm Optimization to Adaptive IIR Phase Equalization," *Proc. of the 2004 ICASSP*, May 2004.

[8] Pal Siddharth, Structured search algorithms for IIR adaptiver and nonlinear filters, Masters Thesis, Pennsylvania State University, July 2004

[9] Tang, K.S., Man, K.F., He, Q. "Genetic Algorithms and their Applications," *IEEE Signal Processing Magazine*, pp. 22-37, November 1996.

[10] Yao L., Sethares W.A., "Nonlinear Parameter Estimation via the Genetic Algorithm," *IEEE Trans. on Sig*. Proc., vol.42, April 1994.

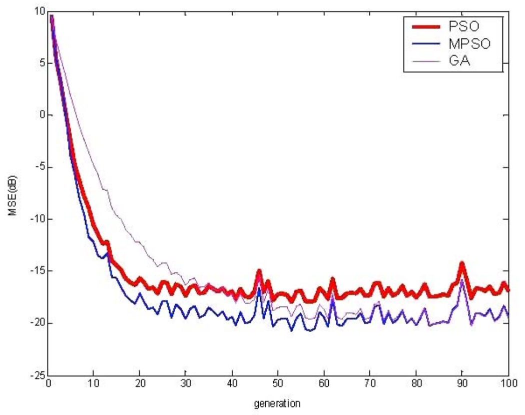


Figure 1. Performance of stochastic search algorithms for Example 1. (Population = 25)

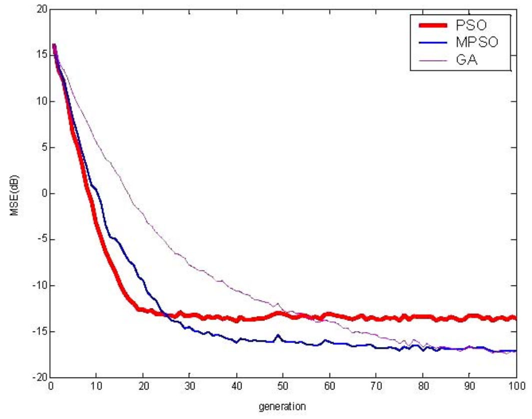


Figure 4. Performance of stochastic search algorithms for Example 2. (Population = 25)

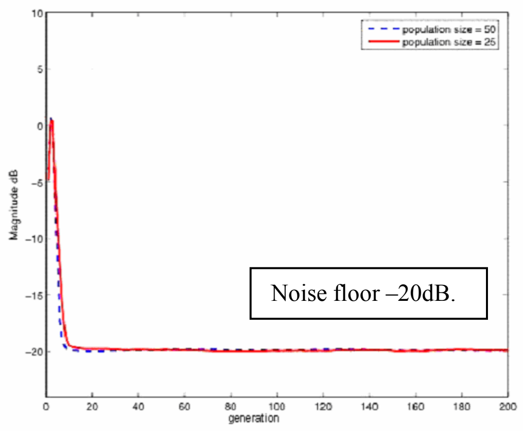


Figure 2. Performance of CRS algorithm for Example 1.

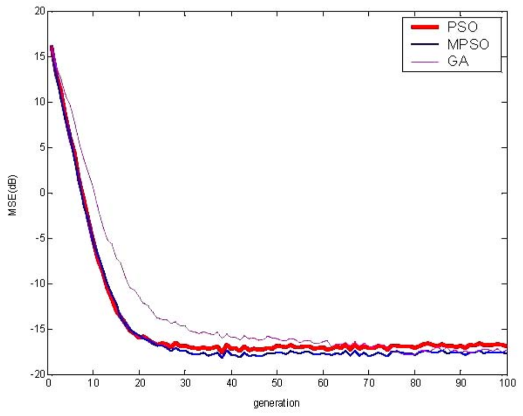


Figure 5. Performance of stochastic search algorithms for Example 2. (Population = 50)

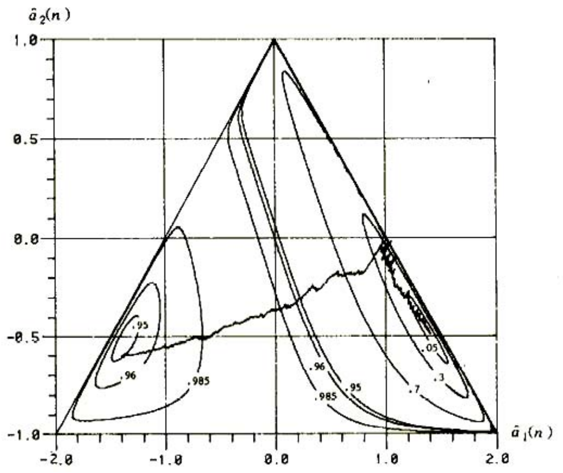


Figure 3. Mean squared error surface for Example 2.

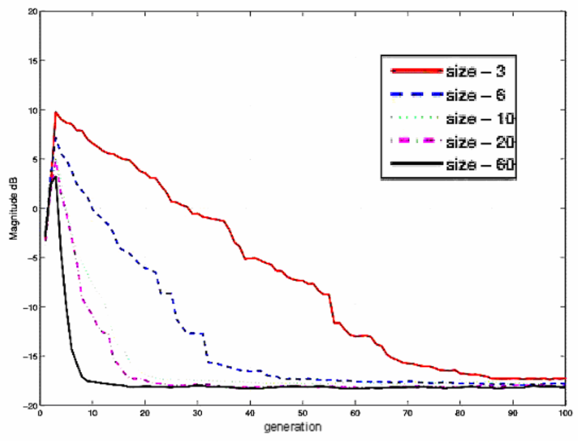


Figure 6. Performance of the CRS algorithm for Example 2