Publication P1

J. Martikainen and S. J. Ovaska
"Designing multiplicative general parameter filters using adaptive genetic algorithms"
in
Proc. of the Genetic and Evolutionary Computation Conference
Seattle, WA, 2004, pp. 1162-1167.

Designing Multiplicative General Parameter Filters Using Adaptive Genetic Algorithms

Jarno Martikainen[1], Seppo J. Ovaska[2]

Institute of Intelligent Power Electronics, Helsinki University of Technology,
P.O. Box 3000, FIN-02015 HUT, Finland
[1]jkmartik@cc.hut.fi, [2]ovaska@ieee.org

Abstract. Multiplicative general parameter (MGP) approach to finite impulse response (FIR) filtering introduces a novel way to realize cost effective adaptive filters in compact very large scale integrated circuit (VLSI) implementations used for example in mobile devices. MGP-filter structure comprises of additions and only a small number of multiplications, thus making the structure very simple. Only a couple of papers have been published on this recent innovation and, moreover, MGP-filters have never been designed using adaptive genetic algorithms (GA). The notion suggesting the use of adaptive parameters is that optimal parameters of an algorithm may change during the optimization process, and thus it is difficult to define parameters beforehand that would produce competitive solutions. In this paper, we present results of designing MGP-FIR basis filters using different types of adaptive genetic algorithms, and compare the results to the ones obtained using a simple GA.

1 Introduction

Predictive lowpass and bandpass filters play an important role in numerous delay-constrained signal processing applications, especially in the area of 50/60 Hz power systems instrumentation. In these applications distorted line voltages or currents should be filtered without delaying the fundamental frequency component. Vainio and Ovaska introduced the multiplicative general parameter (MGP) finite impulse response (FIR) filtering scheme in [1] and [2]. Since the line frequency tends to vary within a constrained interval, typically ±2%, adaptive filters should be used. In MGP-FIR the adaptation is achieved through adjusting the two MGPs. The coefficient values of the basis filter do not change during the adaptation process. An MGP-FIR is designed in the following way. First we optimize the basis filter either by applying traditional hard computing methods, or, as this paper presents adaptive genetic algorithms [5]. Next, the MGP-FIR is used in the actual application, and fine-tuning is left to the multiplicative general parameters. The better the basis filter attenuates unwanted components and the better the prediction capabilities of the basis filter are, the easier it is for the MGP filter to adapt to the changing characteristics of the input signal. Digital filters have been designed before using genetic algorithms, for example in [8] [9], but MGP-FIRs have never been designed using adaptive genetic algorithms. Thus, the speedup in convergence of the GA in this application using adaptive parameters was yet to be studied.

The paper is organized as follows. Section 2 describes the multiplicative general parameter approach to filtering. Section 3 explains the basics of genetic algorithms and defines their use in context with MGP filtering. Section 4 introduces the obtained results. Section 5 includes concluding remarks and Section 6 presents the paths of our future research work.

2 Multiplicative General Parameters

The MGP filtering method sets low computational requirements to the implementation platform, while it simultaneously provides effective and robust filtering of disturbances. All this is achieved without delaying the primary signal component, and maintaining adaptation capabilities around the nominal frequency.

In a typical MGP-FIR, the filter output is computed as

$$y(n) = g_1(n)\sum_{k=0}^{N-1} h_1(k)x(n-k) + g_2(n)\sum_{k=0}^{N-1} h_2(k)x(n-k). \quad (1)$$

Where $g_1(n)$ and $g_2(n)$ present the MGP parameters, and $h_1(k)$ and $h_2(k)$ are the fixed coefficients of an FIR basis filter. Taking this into account, the coefficients of the composite filter are $\theta_1(k) = g_1(n)h_1(k),\ k \in [0,1,...,N-1]$, for the first MGP, and $\theta_2(k) = g_2(n)h_2(k),\ k \in [0,1...,N-1]$, for the second MGP. An example of MGP-FIR with *N*=4 is shown in Fig. 1. Here *N* denotes the filter length.

The filter coefficients in the adaptation process are updated as follows.

$$g_1(n+1) = g_1(n) + \mu e(n)\sum_{k=0}^{N-1} h_1(k)x(n-k) \quad (2)$$

$$g_2(n+1) = g_2(n) + \mu e(n)\sum_{k=0}^{N-1} h_2(k)x(n-k). \quad (3)$$

Where μ is the adaptation gain factor and $e(n)$ is the prediction error between the filter output and the training signal. The MGP-FIR has two adaptive parameters to adapt only to the phase and amplitude of the nominal frequency. More degrees of freedom would allow the filter to adapt to undesired properties, such as the harmonic frequencies. The training signal $s(t)$ is defined as

$$\begin{aligned} s(n) &= \sin(2\cdot\pi\cdot 49\cdot n) + \sum_{m=3,5,7,...}^{15} 0.1\cdot\sin(2\cdot\pi\cdot m\cdot 49\cdot n) + 0.004\cdot r(n), 0 < n \le 300 \text{ samples} \\ s(n) &= \sin(2\cdot\pi\cdot 50\cdot n) + \sum_{m=3,5,7,...}^{15} 0.1\cdot\sin(2\cdot\pi\cdot m\cdot 50\cdot n) + 0.004\cdot r(n), 300 < n \le 600 \text{ samples} \\ s(n) &= \sin(2\cdot\pi\cdot 51\cdot n) + \sum_{m=3,5,7,...}^{15} 0.1\cdot\sin(2\cdot\pi\cdot m\cdot 51\cdot n) + 0.004\cdot r(n), 600 < n \le 900 \text{ samples}. \end{aligned} \quad (4)$$

r(*n*) denotes a uniformly distributed random value between -1 and 1. The duration of the training signal is 900 samples. This signal is divided into three parts, each of which contains 300 samples. These training signal blocks correspond to frequencies of 49 Hz, 50 Hz, and 51 Hz. The training signal constitutes thus of the nominal frequency sinusoid, odd harmonics up to the 15th with amplitudes 0.1 each, and white noise. The training signal is similar to that used in [1] and [2], and it mimics the line signal (voltage/current) with varying fundamental frequency, harmonics, and noise.

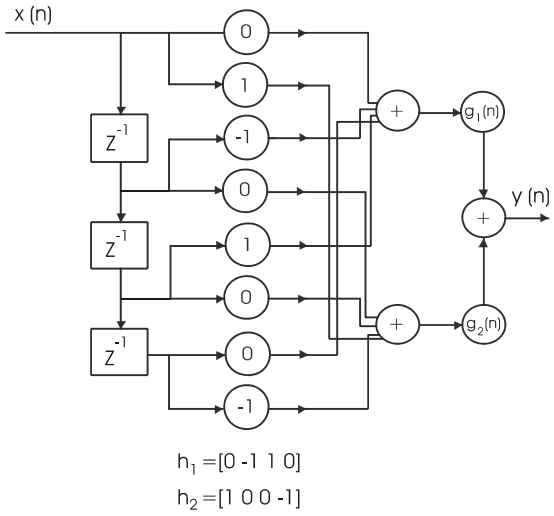

Fig. 1. An example of MGP implementation, where *N*=4. Signal values $x(n-1)$ and $x(n-2)$ are connected to the first MGP and values $x(n)$ and $x(n-3)$ are connected to the second MGP with filter coefficients -1, 1, 1, and -1 respectively

The basic idea of MGP-FIR filters is that all the samples of input delay line should be connected either to the first or to the second MGP, and no value should be left unused. Computational efficiency of these particular MGP filters arises from the fact that the filter coefficients are either -1, 0, or 1. Thus the number of multiplications in filtering operations is radically reduced compared to a normal filtering operation using more general coefficient values. Especially in VLSI or other hardware implementations, the number of multiplications, as well as other operations should be kept at minimum to guarantee fast operation and low cost of implementation.

3 Genetic Algorithms

Genetic algorithms model the nature's way of optimizing the characteristics of a system to comply with the demands of the surrounding environment. Genetic algorithms manipulate the possible problem solutions using selection methods, crossover and mutation operators, and survival strategies [3] [4] [7].

In this application, the cross connections and coefficients of the basis filter are to be optimized using a GA. Using non-evolutionary methods would limit the set of tools to using exhaustive search, since no gradient information is available in this problem domain. These filter tap cross connections and coefficients are modeled as chromosomes in the following way. There are two separate vectors in one chromosome: the first vector, $h_1(n)$, of the chromosome represents the weights of the input samples connected to the first MGP, and the second vector, $h_2(n)$, corresponds to the weights of the input samples connected to the second MGP. The weights can be -1, 0, or 1. -1 and 1 in a vector indicate that this particular input value is associated to the corresponding MGP. 0, on the other hand, determines that a particular input value is not associated with the MGP in concern, rather it is connected to the other MGP. An example of a chromosome is presented in Table 1.

Table 1. Example of the the chromosomes, N=40. h_1 and h_2 associate the input signal values to the first and second MGPs, respectively

h_1	1 1 -1 0 0 1 -1 0 1 1 0 -1 -1 0 1 0 -1 -1 0 1 0 0 0 1 1 0 -1 0 -1 0 1 -1 0 0 0 -1 0 1 1 0
h_2	0 0 0 1 -1 0 0 1 0 0 -1 0 0 1 0 -1 0 0 -1 0 1 -1 1 0 0 1 0 1 0 -1 0 0 1 1 1 0 -1 0 0 -1

In [2] Ovaska and Vainio stated that satisfactory results can be achieved using an initial population, the size of which is twice the number of the filter length. Since the filter length N in this case is 40, the initial population size was set to be 80.

In this paper, we compare five different genetic algorithms. The first one is a simple genetic algorithm with no adaptive parameters. The second genetic algorithm applies adaptive mutation and crossover probabilities as presented in [5]. The third genetic algorithm features also the adaptive mutation and crossover scheme, but with different parameters than in [5]. In the fourth genetic algorithm, we used adaptive μ (see Eqs. (2) and (3)). The fifth algorithm used modified probabilities of [5], adaptive μ, and seeding.

The fitness of the chromosomes is evaluated using the fitness function:

$$fitness = \frac{K}{\max(NG49, NG50, NG51) \cdot (ITAE49 + ITAE50 + ITAE51)}. \quad (5)$$

K is assigned a value of 10^7 to scale the output of the fitness function to be expressed in thousands. Terms $NG49$, $NG50$, and $NG51$ represent the noise gain at a specific stage of the test input signal. These terms were added to the fitness function to attenuate noise. The noise gain is calculated as

$$NG(n) = \sum_{k=0}^{N-1} [g_1(n) \cdot h_1(k)]^2 + \sum_{k=0}^{N-1} [g_2(n) \cdot h_2(k)]^2 . \quad (6)$$

$g_1(n)$ and $g_2(n)$ represent the first and second MGP at the end of a certain frequency period, respectively, whereas $h_1(n)$ and $h_2(n)$ denote the filter coefficients associated to the corresponding MGPs. In other words, $NG49$ is calculated using the MGP values after 300 samples, $NG50$ and $NG51$ are calculated after 600 and 900 samples, respectively, the frequency of the training signal changing every 300 samples.

*ITAE*49, *ITAE*50, and *ITAE*51stand for the Integral of Time-weighted Absolute Error (ITAE) for each of the three signal parts, respectively. These terms were added to the fitness function to smoothen the adaptation of the filter to the varying input signal characteristics. The ITAE is calculated as follows.

$$ITAE = \sum_{n=1}^{M} n \cdot e(n). \tag{7}$$

n is the sample index, and *M* stands for the sample number at the end of a specific frequency period, in this case 300, 600 and 900. $e(n)$ is the error between the output of the system and the pure input signal without any harmonics or noise. The GA may in some cases create filters with unreasonably low noise gains, leading thus to very high fitness value of chromosomes. Theoretical minimum limits can be calculated that have to be passed in order the filter to be practical [10]. If these limits were not met, a penalty term was added to corresponding chromosomes' fitness value so that it was very likely eliminated from the next generation.

The genetic algorithms were given 1000 generations to converge to a solution, and each genetic algorithm was run 20 times to get some statistical reliability. Elitist mutation scheme was applied in each case, so that the best chromosome was never mutated. A single simulation run of 1000 generations using initial population of 80 chromosomes and filter length of 40, took 70-80 minutes to run on a Dell computer equipped with 2.66 GHz Intel Pentium 4 processor, 512 MB of memory, and MATLAB 6.5.0 software.

3.1 Reference Genetic Algorithm

To create a reference point, we implemented a simple genetic algorithm without any fine tuning to solve the problem. The algorithm starts by creating the initial population. The probabilities of assigning -1, 0, or 1 to a particular gene in the chromosome related to the first MGP are 1/3 all. In the case of assigning -1 or 1 to a gene in $h_1(n)$, 0 is assigned to this same gene in $h_2(n)$. This is because one input sample can be connected to only one MGP, not both. If 0 is assigned to a gene in the vector related to the first MGP, -1 or 1 is assigned to a gene related to the second MGP with the probability of 1/2 for both -1 and 1.

After the generation of the initial population, the chromosomes are arranged in descending order according to their fitness values. Mating is based purely on the rank of the chromosomes: the best and the second best chromosomes act as parents for two offspring. The offspring are created using single-point crossover, the crossover point of which is selected randomly. Also, the third and fourth best chromosomes constitute parents for two offspring, and so on. Altogether, 40 best chromosomes produce 40 offspring. Thus the 40 chromosomes with the lowest fitness scores are replaced by the offspring of the 40 best chromosomes. Also the parents remain within the population thus keeping the population size constant at 80 chromosomes.

The convergence behavior of this reference GA can be seen in Fig. 2. This figure shows clearly how the simple genetic algorithm converges in a typical optimization run. Maximum and average fitness values for each generation are well separated, and the minimum fitness lies very close to zero. This simple GA does not fully converge during the 1000 generations it was allowed to operate. Mutation probability of 5% and μ value 0.002 were used in this GA.

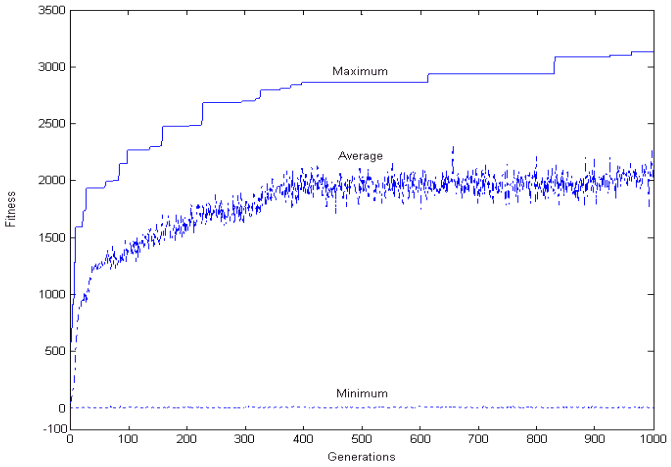

Fig. 2. Typical convergence behavior of the reference GA

3.2 Adaptive Genetic Algorithm

Optimal parameters for a GA may change during the iteration process and thus it is usually difficult to choose initial parameters that produce satisfactory results. Adaptive GA (AGA) parameters have been studied by Srinivas and Patnaik in [5]. In their approach, the mutation probability, p_m, and the crossover probability, p_c, are adjusted as follows.

$$p_c = \frac{k_1\left(f_{\max} - f'\right)}{f_{\max} - f_{ave}}, \quad f' \geq f_{ave} \tag{8}$$

$$p_c = k_3, \quad f' < f_{ave} \tag{9}$$

$$p_m = \frac{k_2\left(f_{\max} - f\right)}{f_{\max} - f_{ave}}, \quad f \geq f_{ave} \tag{10}$$

$$p_m = k_4, \quad f < f_{ave}\,. \tag{11}$$

k_1, k_2, k_3, and k_4 represent coefficients with values equal or less than one. f_{max} denotes the maximum fitness value of a generation, whereas f_{ave} denotes the average fitness value of the same generation. f' denotes the higher fitness value of a parent pair. The authors of [5] propose values 1.0, 0.5, 1.0, and 0.5 for k_1, k_2, k_3, and k_4 respectively.

Using this approach, the crossover probability decreases as the fitness value of the chromosome approaches f_{max} and is 0.0 for a chromosome with fitness value equal to f_{max}. Also, mutation probability approaches zero as the fitness of a chromosome approaches f_{max}. μ value 0.002 was also used in this GA.

The convergence behavior of this AGA can be seen in Fig. 3. This algorithm converges rapidly and the maximum value stays the same for the rest of the time. The maximum is however local, since the fitness value is smaller than with the reference GA. Maximum, average, and minimum fitnesses are the same after about 70 generations.

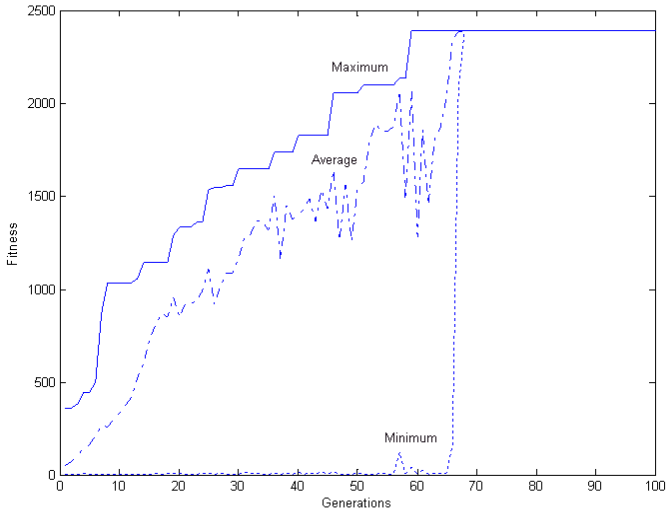

Fig. 3. Typical convergence behavior of the AGA

3.3 Adaptive Genetic Algorithm with Modified Probabilities

The values for k_1, k_2, k_3, and k_4 presented in [5] are not obviously appropriate for all applications. Referring to the "no free lunch" theorem discussed by Wolpert and Macready in [6] and Fogel in [7], we decided to give value equal to one for each of the k_1, k_2, k_3, and k_4, because the values presented in [5] do not perform so well in this application. This way we may prevent premature convergence, which seemed to be a problem in our application, when 1.0, 0.5, 1.0, and 0.5 were assigned to $k_i, i = 1, 2, 3, 4,$ respectively. Since elitist mutation saves the best solution from corruption, we apply the mutation probability of 1.0 to all but the best chromosome. This way, although possibly near the optimal solution, we still search efficiently the surroundings for a better one. Typical convergence behavior of this AGA can be seen in Fig. 4. This GA converges to the maximum rather slowly, while the minimum stays close to 0 all the time.

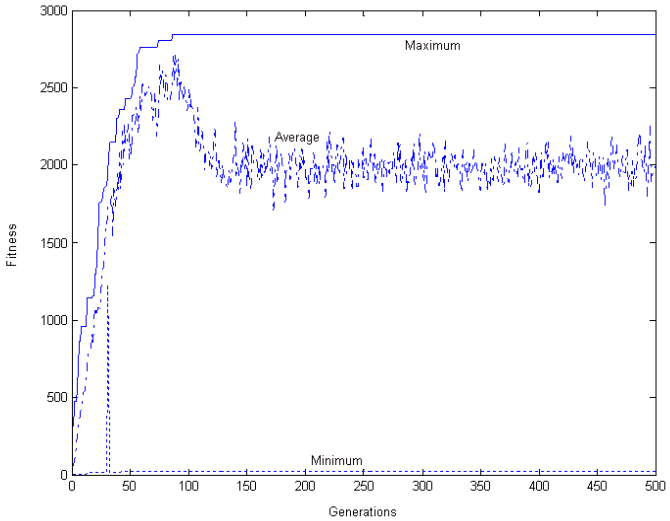

Fig. 4. Typical convergence behavior of the AGA using modified probabilities

3.4 Adaptive Genetic Algorithm with Modified Probabilities and the µ-Term

In the previously presented GAs we used a constant adaptation gain factor μ (see Eqs. (2) and (3)). Since the characteristics of the input signal vary with time, it is difficult to determine the optimal parameters of the system before the optimization process. Adaptive μ was applied in order to speed up the convergence of genetic algorithm.

The chromosome was modified so that one gene was added to present the μ value. The μ values in the initial population were chosen randomly so that $0 < \mu < 0.009$. Values larger than this can easily make the filter unstable. In crossover the offspring μ value, μ_o, was calculated as follows.

$$\mu_o = r \cdot \mu_{p1} + (1 - r) \cdot \mu_{p2} . \quad (12)$$

r stands for a random number between 0 and 1. μ_{p1} and μ_{p2} are the μvalues of the first and second parents, respectively. Mutation was applied as follows.

$$\mu_m = \mu + 0.1\mu, r < 0.5 \quad (13)$$
$$\mu_m = \mu - 0.1\mu, r \geq 0.5 .$$

μ_m presents the value of μafter mutation, and r is a uniformly distributed random number between 0 and 1. An example of the converged μ values and corresponding fitnesses are shown in Table 2. The average converged μ value of the 20 runs was 0.0034. This value can be considered as the practical maximum μ value. Values larger than 0.0034 could easily cause the filter to be unstable, whereas values below 0.0034 merely slow the convergence of the GA down.

Table 2. Example of converged μ values an correponding fitnesses. Each run was done using 1000 generations

Run	Converged μ	Best fitness
1	0.0034	3420
2	0.0025	3069
3	0.0032	3735
4	0.0030	3225
5	0.0042	3387
6	0.0044	2962
7	0.0034	2877
8	0.0042	3249
9	0.0029	3504
10	0.0036	3052
11	0.0029	3541
12	0.0033	3447
13	0.0037	3761
14	0.0026	3606
15	0.0039	3481
16	0.0040	3215
17	0.0037	3304
18	0.0026	3776
19	0.0028	3772
20	0.0026	3722
Average	0.0034	3405

The convergence of this AGA with modified probabilities of [5] and adaptive μ is shown in Fig. 5. This GA converges quite rapidly, and the average fitness follows quite close the maximum fitness. The minimum fitness values vary a lot, because μ is a sensitive parameter and even small changes can easily make the filter unstable resulting in a low fitness value.

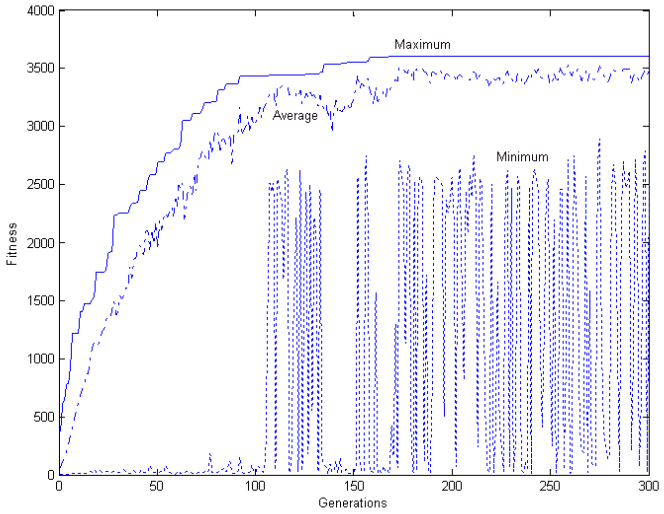

Fig. 5. Typical convergence behavior of the AGA using adaptive μ

3.5 Adaptive Genetic Algorithm Using Seeding

When analyzing the chromosomes produced by the GA mentioned above, one could see a systematical structure in the converged solutions. All the GAs had a tendency of creating blocks of consecutive equal gene values within a chromosome. An example of a converged chromosome can be seen in Table 3.

Table 3. Example of converger chromosomes. Note the block like structure of the genes

h_1	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 0 0
h_2	-1 1 1 1 -1 1 1 1 -1 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 1 0 0 0 0 0 0 0 0 0 0 -1 1 1 1

This kind of repeating structure was present in nearly all the converged chromosomes. Bearing this in mind, seeding was applied to the GA while generating the initial population. Seeding means applying the information of the structure of the solution to the generation process of the initial set of chromosomes. Seeding was implemented so that the chromosomes in the initial population we formed of blocks of random number of consecutive equal gene values. The block length was set to be 1 at the minimum and the other extreme was a chromosome constituting only of a single gene value.

The typical convergence behavior of this GA is presented in Fig. 6. The convergence rate is rather fast. Variations in minimum fitness are explained by the adaptive μ, since the filter can easily become unstable if the μ value changes too drastically.

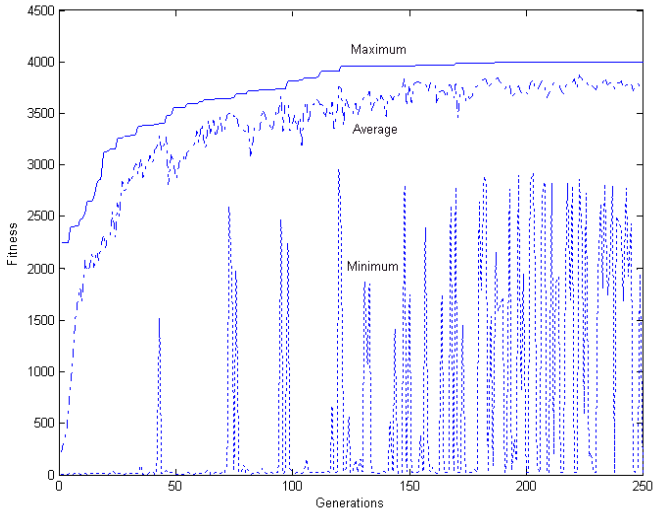

Fig. 6. Typical convergence behavior of the AGA using modified probabilities, adaptive μ, and seeding

4 Results

The performance of the different GA approaches in MGP filter design are reported in Table 4. Each GA took about 70-80 minutes to go through 1000 generations. However, there were significant differences in the convergence properties of the algorithms. The adaptive GA proposed in [5] was clearly the fastest to converge, but the algorithm was very likely trapped in a local minimum, and it thus behaved worse in statistical sense than the simple GA used for reference.

Modifying the probabilities of the AGA produced fitness average closer to the reference GA than with original parameters, while simultaneously reducing the number of required generations to one tenth. Adding the adaptive μ term succeeded in achieving higher average fitness, while simultaneously increasing the number of required generations. Applying seeding helped the adaptive μ enhanced AGA to produce best results, indicating highest average fitness with even better convergence characteristics as the AGA with adaptive μ but without seeding. Standard deviations presented in Table 4 indicate that the AGA with modified probabilities, adaptive μ, and seeding converges to high fitness solutions rather reliably.

The time to achieve the best results was 11.5 minutes, which is about 18% of the time required by the reference GA to produce results. The gained speedup makes the introduced method a tempting tool for practicing engineers. The average time to go through 1000 generations is almost the same for all the GAs. Then again, the differences between the average convergence times are notable.

Table 4. Performance comparison of the featured Gas. The results are averages of 20 runs

Genetic Algorithm	Average Best Fitness / Standard Deviation	Convergence (generations) / Standard Deviation	Time (mins)	Convergence Time (mins)
Simple GA	3249 / 324	837 / 130	76.4	63.9
AGA	1472 / 505	21 / 11	77.9	1.6
AGA, Modified Probabilities	2824 / 557	89 / 34	76.7	6.8
AGA, Modified Probabilities and μ	3638 / 287	329 / 218	73.9	24.3
AGA, Modified Probabilities, μ, seeding	3617 / 329	156 / 65	73.6	11.5

The best converged chromosome had fitness value of 3985 and was achieved using AGA with adaptive μ and seeding. This chromosome is presented in Table 5.

Table 5. The best converged chromosome

h_1	1 1 1 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0
h_2	0 0 0 -1 -1 -1 -1 -1 -1 -1 0 0 0 0 0 0 0 0 0 0 1 -1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1

Fig. 9 describes the signals before and after filtering. The evaluation signal *eval*(*n*) used is defined as follows.

$$eval(n) = \sin(2\cdot\pi\cdot 49\cdot n) + \sum_{m=3,5,7,\ldots}^{15} 0.15\cdot\sin(2\cdot\pi\cdot m\cdot 49\cdot n),\ 0 < n \le 300\ \text{samples} \tag{14}$$

$$eval(n) = \sin(2\cdot\pi\cdot 50\cdot n) + \sum_{m=3,5,7,\ldots}^{15} 0.15\cdot\sin(2\cdot\pi\cdot m\cdot 50\cdot n),\ 300 < n \le 600\ \text{samples}$$

$$eval(n) = \sin(2\cdot\pi\cdot 51\cdot n) + \sum_{m=3,5,7,\ldots}^{15} 0.15\cdot\sin(2\cdot\pi\cdot m\cdot 51\cdot n),\ 600 < n \le 900\ \text{samples}.$$

The amplitude response of the filter presented in Table 5. after 600 samples of the evaluation signal can be seen in Figs. 7. This response shows how the GA tries to dampen the harmonic frequencies of the nominal components (i.e., 150 Hz, 250 Hz, …, 750 Hz). The phase delay in samples is shown in Fig. 8. The phase delay is negative in the 50 Hz region, and thus the filter has the desired predictive capabilities. Table 6 shows the calculated total harmonic distortion (THD) values for each of the training signal nominal frequency components before and after filtering. After filtering the THD values as well as the amplitudes of the harmonics are about a tenth of the values before filtering.

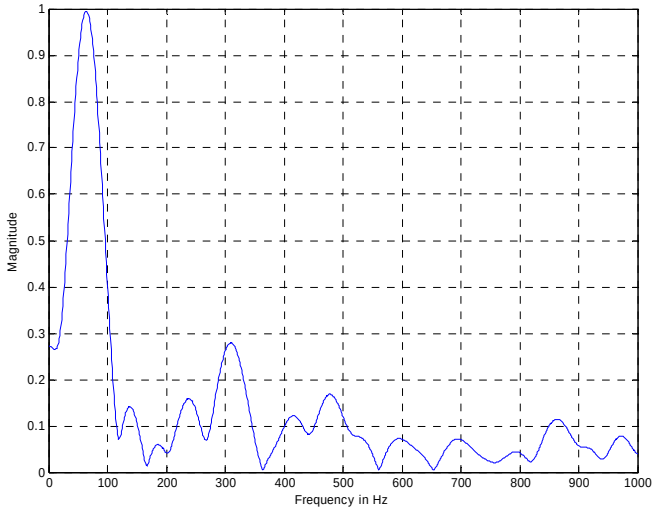

Fig. 7. Instantaneous magnitude responses of the MGP-FIR

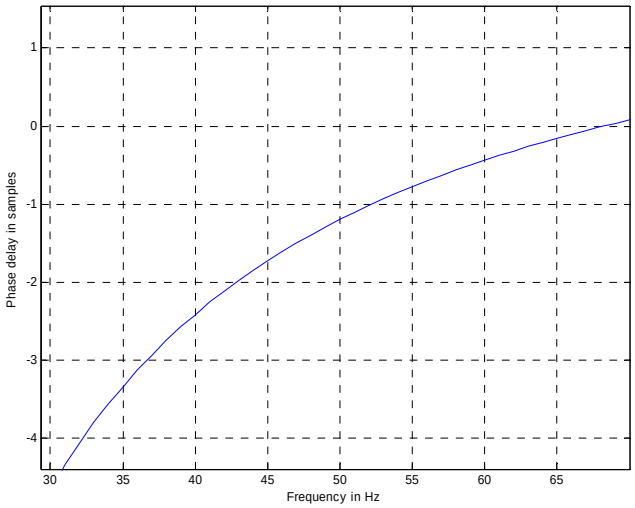

Fig. 8. Phase delay of the MGP-FIR. The phase delay in samples is negative in the 50 Hz region, as it should be in predictive filters

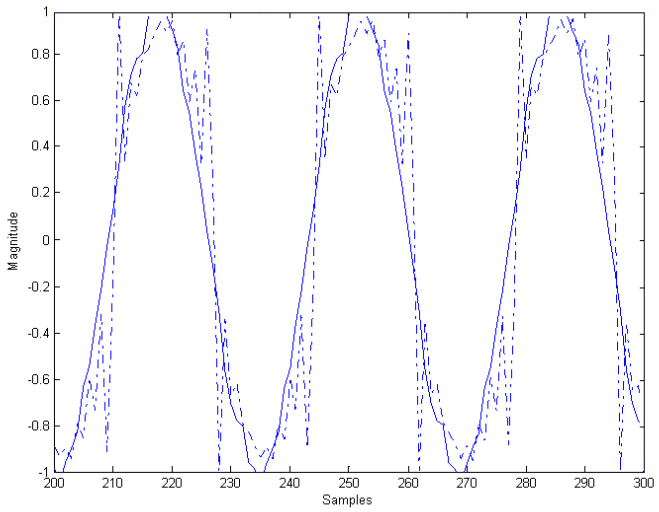

Fig. 9. Test signal defined in Eq. (13) (dash dotted) is filtered using the best MGP-FIR given in Table 5. Output (solid line) is almost sinusoidal

Table 6. Total harmonic distortions (THD) of the best MGP-FIR basis filter

	Input amplitude	Output at 49 Hz	Output at 50 Hz	Output at 51 Hz
Nominal frequency	1	0.87	0.93	0.88
3^{rd} harmonic	0.15	0.018	0.016	0.011
5^{th} harmonic	0.15	0.023	0.025	0.023
7^{th} harmonic	0.15	0.018	0.009	0.008
9^{th} harmonic	0.15	0.012	0.012	0.013
11^{th} harmonic	0.15	0.010	0.008	0.006
13^{th} harmonic	0.15	0.005	0.002	0.007
15^{th} harmonic	0.15	0.005	0.004	0.007
THD	39.7%	4.4%	3.7%	3.6%

5 Conclusions

In this paper, we have presented a new way to design MGP-FIR basis filters. Adaptive genetic algorithms produced better results than genetic algorithms without any adaptive parameters. Knowing what to look for, or understanding the structure of the possible solution helped to apply seeding in the GA in order to further speed up the optimization process. Also, it is likely that using adaptive μ and seeding as parts of the design process had a greater impact on the final results than the use of adaptive mutation and crossover probabilities. MGP-FIRs were designed using evolutionary programming in [2], but the approaches used in the present paper make the GA converge more rapidly. Having said that, we conclude that the best proposed method excels among the others in MGP-FIR design methods, and should therefore be regarded as a competitive tool for experts working in the field of power systems instrumentation.

6 Future Work

Our future work will concentrate on experimenting how multiple population GAs would affect the performance of the optimization process. Also, since GAs are by nature well parallelizable, a parallel computing approach to GA aided design of MGP-FIRs will be explored.

References

1. Vainio, O., Ovaska, S.J., Pöllä, M.: Adaptive filtering using multiplicative general parameters for zero-crossing detection. IEEE Transactions on Industrial Electronics, Vol. 50, No. 6, Dec 2003, 1340-1342
2. Ovaska, S.J, Vainio, O.: Evolutionary programming in the design of adaptive filters for power systems harmonies reduction. IEEE International Conference on Systems, Man and Cybernetics, Vol. 5, Oct 2003, 4760-4766
3. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York, NY (1996)
4. Haupt, R. L., Haupt, S. E.: Practical Genetic Algorithms. John Wiley & Sons, New York, NY (1998)
5. Srinivas, M., Patnaik, L. M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 4, Apr 1994, 656-667
6. Wolpert, D. H., Macready, W. G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, Apr 1997
7. Fogel, D.B.: Evolutionary Computation, Toward a New Philosophy of Machine Learning. IEEE Press, Piscataway, NJ (2000)
8. Wade, G., Roberts, A., Williams, G.: Multiplier-less FIR filter design using a genetic algorithm. IEE Proceedings of Vision, Image and Signal Processing, Vol. 143, No. 3, Jun 1996
9. Lee, A.; Ahmadi, M., Jullien, G.A., Miller, W.C., Lashkari, R.S.: Digital filter design using genetic algorithm. IEEE Symposium on Advances in Digital Filtering and Signal Processing, 1998, Jun 1998, 34-38
10. Vainio, O., Ovaska, S.J.: Noise reduction in zero crossing detection by predictive digital filtering. IEEE Transactions on Industrial Electronics, Vol. 42, No. 1, Feb 1995, 58-62