

Digital Filter Design using Multiple Pareto Fronts

Thorsten Schnier and Xin Yao

School of Computer Science, The University of Birmingham

Edgbaston, Birmingham B15 2TT, UK

Email: {T.Schnier,X.Yao}@cs.bham.ac.uk

Pin Liu

Network Strategy, Broadband Routing & Switching Division

Marconi Communications, Discovery Court, 551-553 Wallisdown Road

Poole, Dorset BH12 5AG, UK

Email: pin.liu@marconi.com

Abstract

Evolutionary approaches have been used in a large variety of design domains, from aircraft engineering to the designs of analog filters. Many of these approaches use measures to improve the variety of solutions in the population. In this paper, clustering and Pareto optimization are combined into a single evolutionary design algorithm. The objective of this is to prevent the system from converging prematurely to a local minimum and to encourage a number of different designs that fulfill the design criteria. Our approach is demonstrated in the domain of digital filter design. Using a polar coordinate based pole-zero representation, two different lowpass filter design problems are explored. The results are compared to designs created by a human expert. The results demonstrate that the evolutionary process is able to create designs that are competitive with those created using a conventional design process by a human expert. They also demonstrate that each evolutionary run can produce a number of different designs with similar fitness values, but very different characteristics.

1 Introduction

Evolvable hardware (EHW) refers to one particular type of hardware whose architecture/structure and functions change dynamically and autonomously in order to improve its performance in performing certain tasks [1, 2]. The emergence of this new field in recent years has been influenced profoundly by the progresses in reconfigurable hardware and evolutionary computation. Traditional hardware is notorious for its inflexibility. It is impossible to change the hardware structure and its functions once it is made. However, most real world problems are not fixed. They change with time. In order to deal with these problems efficiently and effectively, different hardware structures are necessary. EHW provides an ideal approach to make hardware “soft” by adapting the hardware structure to a problem dynamically.

This paper describes an evolutionary design approach to digital filter design. We have applied three different methods to achieve population diversity in our research, i.e., Pareto optimisation, fitness sharing and clustering. Our empirical studies show that the new approach can lead to improved designs of different characteristics.

The rest of this paper is organised as follows. Section 2 introduces the basics about digital filter design. Section 3 describes our evolutionary approach. Section 4 presents the experimental results. Finally, Section 5 concludes the paper with a brief summary and a few remarks.

2 Digital Filter Design

Digital filter design is a task with engineering relevance [3]. The two design problems considered in this paper were first studied by Lu [3].

In general, digital filter design is usually a two-step process. In the first step, a mathematical description of the filter fulfilling the design criteria is derived. This description is then transformed into a hardware description in the second step. The two steps are very different in terms of difficulty, methods employed, and performance criteria. Similar to the most filter design papers [3], our work reported here only deals with the more difficult first step. It produces an optimal (near optimal) polynomial that can then be transformed into hardware implementation.

Any linear digital filter can be mathematically specified by a complex-numbered polynomial function, i.e., the transfer function (Equation 1). This polynomial function, i.e., Equation 2, can be rewritten as the quotient of two product terms with the numerator specifying the zeroes of the polynomial and the denominator specifying the poles. The function usually has a scaling constant (Equation 2). The two descriptions are equivalent. It is easy to transform a pole-zero description to a polynomial description, but not vice versa. The frequency response can be derived from the transfer function by calculating the values for $z = e^{j\omega T}$ with T the sampling frequency.

$$H(z) = \frac{\sum_{i=0}^n b_i z^{n-i}}{z^{(n-r)} \sum_{i=0}^r b_i z^{r-i}} \quad (1)$$

$$H(z) = b_0 * \frac{(z - z_{z0})(z - z_{z1}) \dots (z - z_{zn})}{z^{r-i}(z - z_{p0})(z - z_{p1}) \dots (z - z_{pi})} \quad (2)$$

Not all transfer functions can be realized in a hardware filter. Two main requirements have to be observed:

Real coefficients: The coefficients in the polynomial description have to be real numbers. In terms of poles and zeroes, this can be achieved if all poles and zeroes are either real, or exist in conjugate-complex pairs (i.e. $a + jb$ and $a - jb$).

Stability: In a stable filter, a bounded input will always produce a bounded output. A filter is only stable if all poles are within the unit circle, i.e. $\|a + jb\| < 1$. While there are uses for unstable filters in specific applications, most filters are designed to be stable.

Filter performance is usually multi-objective. The two filters considered in this report are low-pass filters. An ideal low-pass filter lets signals pass unchanged in the lower frequency region (passband), and blocks signals completely in the upper frequency region (stopband). In reality, a transition band is often located between passband and stopband.

To minimise distortion in the signal in the passband, two criteria have to be met. The first is that the amplitude of the frequency response in the passband should be as constant as possible. The second criterion is that the phase in the passband has to be as linear as possible. In practice, the so-called ‘group delay,’ i.e., the first derivative of the phase $\delta\phi/\delta\omega$, is often used. The second criterion can therefore be stated as a constant group delay. This means that all frequencies are delayed by the filter by the same amount of time.

In the stopband, the design goal is generally to attenuate the signal as much as possible. Because the signal is attenuated, the phase and group delay of the signal in the stopband usually becomes unimportant.

Figure 1 shows a typical lowpass filter. The top half shows the amplitude and the lower half the group delay. The ideal behaviour is shown with thick lines, the ‘real’ behaviour (thin line) is acceptable as long as it is within the shaded regions.

To facilitate comparison between our and other existing work, this paper follows the criteria used previously [3] whenever possible, that is,

1. weighted square error over the amplitude in the passband and stopband,
2. peak amplitude in the stopband,
3. maximum deviation from constant amplitude in the passband,
4. maximum deviation from the goal group delay, and
5. the stability.

The conventional design process adopted by human experts uses criterion (1) as optimisation criterion and criteria (2) to (5) with predefined values as constraints.

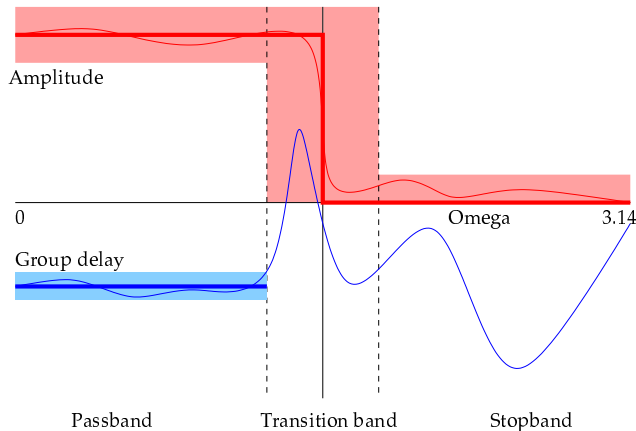


Figure 1: A lowpass filter which is used as the first test problem in this report.

3 The Evolutionary Approach

As described in the previous section, the transfer function is generally given in one of two forms: a polynomial or a pole-zero description of the filter. Because of the direct relationship between the transfer function and frequency response, poles and zeroes in the pole-zero form of the transfer function (Equation 2) can be directly interpreted: a pole near the current frequency amplifies the signal, a zero attenuates it. Since poles and zeroes are complex numbers, their locations in the complex plane can be naturally expressed in polar coordinates. Under such coordinates, the angle directly specifies the frequency at which the pole or zero is active, and the distance from the origin indicates its 'strength'.

A polar coordinate based representation of poles and zeroes has number of advantages. Most importantly, it can represent all linear IIR filters, and it is possible (as shown below) to ensure the feasibility of all phenotypes. Additionally, locality is preserved at least on a local scale. That is, similar genotypes will have similar frequency responses. Because of this, the fitness landscape is also fairly smooth.

The transfer function of a filter can be represented by a sequence of paired real-value numbers, where each pair indicates the polar coordinates of a pole or zero. An additional pair of real-valued numbers encode the scaling parameter b_0 . Each pair of real-valued numbers is called a gene in our study.

In order to impose the constraints specified in the previous section, poles and zeroes need to be either positioned on the real axis (i.e. $IM(z) = 0$), or exist in conjugate-complex pairs when a genotype is mapped to a phenotype (i.e., a transfer function). All poles have to be located within a unit circle in order to ensure filter stability.

For example, a genotype of $N_{p2} + N_{p1} + N_{z2} + N_{z1} + 1$ pairs of real-valued numbers consists of

N_{p2} **pole pairs:** Each pair of real valued numbers in the genotype represents a complex pole. The conjugate-complex pole is automatically generated by the genotype-phenotype mapping to ensure the filter is feasible. The radius can lie between -1.0 and 1.0, which ensures stability.

N_{p1} **single poles:** For these poles, the angle is ignored. Only the radius is used to determine the position on the real axis. Radius is restricted to between -1.0 and 1.0.

N_{z2} **zero pairs:** determines one of conjugate-complex pair of zeroes, the partner is automatically generated. The radius can lie between -1.0 and 1.0, but is scaled in the genotype-phenotype mapping with the factor R_{zMax} .

N_{z1} **single zeroes:** The angle is ignored. The radius is between -1.0 and 1.0 and scaled with R_{zMax} .

Scaling factor b_0 : The angle is ignored. The radius is used to scale the polynomial and is between -1.0 and 1.0 and scaled with S_{Max} .

Additionally, the transfer function can have $N(p0)$ poles in the origin. Table 1 shows an example of genotypes with two conjugate-complex zero pairs, a single real zero, a single conjugate-complex pole pair, and a single real pole. It has two additional poles at the origin.

$N_{z2} = 2$		$N_{z1} = 1$		$N_{p2} = 1$		$N_{p1} = 1$		b_0			
1.2	0.9	1.9	0.6	0.5	-0.4	4.1	0.7	2.0	0.5	1.7	0.3

Table 1: An example of genotypes used in our work.

The scaling employed for zeroes and b_0 means that all pairs of real value numbers have exactly the same range: between -1.0 and 1.0 for radius, and between $-\pi$ and π for the angle. This often facilitates evolutionary search without special knowledge about the differences between zeroes, poles, and b_0 . The scales R_{zMax} and S_{Max} are fixed in our experiments. Setting them optimally requires some domain knowledge (see also Section 5).

3.1 Clustering and Identification of Pareto Fronts

The clustering replacement strategy consists of four steps. In the first step, individuals are assigned to one of the clusters. Non-dominated individuals are identified. In the next two steps, a decision will be made on which of these non-dominated individuals will survive to the next generation. Finally, any remaining places in the next generation will be filled up from the remaining individuals in the pool of all parents and offspring.

Recent research has demonstrated the advantage of elitism in Pareto optimisation [4, 5]. In elitist Pareto optimisation, members of the Pareto front will be moved into an external ‘store’ and will not have to compete with other individuals for survival. We have implemented a simplified version of this. Members of the Pareto front remain part of the population in our case, but will be marked as non-dominated. As a consequence, most of them will automatically survive into the next generation.

Because of clustering, more than one Pareto front exists in a population in our evolutionary system. The motivation behind clustering is to maintain multiple Pareto fronts so that they do not compete against each other. The first step of our replacement strategy is to ensure that each individual is assigned to one of the clusters.

In our current implementation, the number of clusters is fixed over the whole run (however, some clusters may be empty, see below). Each single run of the algorithm has three phases, in which clustering is performed in a different way.

Initial phase: After the initial population is generated, it is clustered using the k -means clustering algorithm [6]. As a result of k -means clustering, a cluster-centre is established for each cluster.

Exploration phase: The aim of this phase is to identify a sufficient number of different clusters that have some chance of producing interesting results. In this phase, a population is reclustered at least every n generations ($n = 100$ for most of our experiments). Offspring that are created by crossover between parents from the same cluster or by mutation of a single parent will be assigned to the same cluster as the parent(s). Other offspring will be assigned to the cluster whose centre is closest to it. If the distance to the closest centre is more than m (m is around 1.8 for most of our experiments) times the largest distance between any two cluster centres in the population, a complete re-clustering is triggered. This phase lasts a pre-set number of generations (e.g. 1000).

Exploitation phase: Reclustering and intra-cluster crossover can result in genetic material from one cluster ‘contaminating’ other clusters. If this material is very successful, it could eventually lead to all clusters converging. This does not help the discovery of different design solutions. Therefore, in the exploitation phase, all clusters are frozen. Crossover is only allowed between parents from the same cluster. No reclustering is performed. As a result, there is no interchange of genetic information among the clusters. All new offspring will inherit the cluster information from their parents. It is of course possible, and generally can be observed from our experiments, that within each cluster very different genotypes emerge. Such different genotypes are most likely to populate different areas of the cluster Pareto front.

After all offspring have been assigned to a cluster, the new Pareto fronts are computed. This is done on a per-cluster base. First, the non-dominated individuals in the offspring are computed. Then these are merged with the previous non-dominated individuals. Individuals are also checked against the current fitness constraints as described in Section 3.1.2.

3.1.1 Shrinking Pareto Fronts

In most implementations of Pareto selection with elitism, the number of individuals in the Pareto front will grow continuously through the run. It is necessary to periodically remove individuals from the Pareto front to prevent the front and the population from overgrowing. Individuals are removed from the Pareto front of a cluster whenever the number of individuals in the front is above a pre-set threshold value.

Generally speaking, individuals should be removed whenever many very similar individuals can be found, because there is little incentive to keep very similar individuals. To achieve this, all individuals are ‘paired’: the two individuals which have the smallest genotypic distance are paired, then the two individuals with the next smallest distance are paired, etc. Each individual is allowed to be in only one such pair. Within each pair, one individual is removed. We can repeat this process until sufficient number of individuals have been removed. The number of individuals to be removed is a pre-set parameter in our system.

The decision about which individual of a pair to remove is based on the combined fitness of an individual. The better individual survives. The best individual in a population will never be removed from the population in the shrinking operation.

3.1.2 Tightening Constraints

Using Pareto selection, individuals with poor fitness values can still survive as long as they are not dominated by any other individuals. For example, the final population can easily contain ‘filters’ that do not allow any signal to pass, because this maximises the signal suppression in the stopband.

It is useful to have some kind of constraints for the individuals to restrict the number of such extreme individuals in a Pareto front. Unfortunately, early in the run a large fraction of individuals will violate the constraints. To get around this problem, we have used a self-adaptive mechanism that adapts a constraint vector dynamically to the currently achieved fitness values in the population.

For each particular fitness criterion, three different groups of individuals are distinguished:

- the individuals with the best value for this particular fitness,
- the individuals with the best overall combined fitness, and
- the individuals with the worst value for this particular fitness.

When all fitness criteria are considered, individuals can be in more than one group. In particular, individuals in the first group for one fitness criterion are often also in the third group for a different fitness criterion. For example, individuals with the best signal attenuation in the stopband could have the largest amplitude deviation in the passband. The second group is the most important as it is most promising in producing solutions with useful compromises.

The algorithm used to concentrate evolutionary search on individuals in the second group is guided by two vectors, which have as many elements as there are fitness values. The ‘final constraints vector’ (FCV) is pre-defined and does not change, its values are the constraints that are applied at the end of a run. Typically, the current final constraints are set to be about 3-4 times the expected ‘best individual’ performance. The ‘current constraints vector’ (CCV) is initialised with the worst possible fitness values (positive infinity in the case of minimisation). Every n generations, the CCV is updated and individuals removed:

1. Find the worst individual for each of the fitness criteria.
2. For each of the fitness criteria, calculate the quotient of the value in the FCV to the current worst value in the population. The criterion that has the largest quotient is selected for tightening.
3. Sort all non-dominated individuals according to the criterion selected in the previous step.
4. Identify the worst n individuals and mark them for deletion (n is a pre-defined parameter, e.g. 0.5% for a population of size 1400).
5. Update the CCV by setting the value for the selected criterion to that achieved by the worst remaining individual.

The CCV is used in deciding which individuals should be in a Pareto front. An individual whose one or more fitness values are worse than the values in the CCV will be allowed in the population, but not marked as non-dominated. It will not become part of the Pareto front. The values in the CCV will shrink each time the above algorithm is run. The speed of shrinking depends on the progress of the evolution. Once a value in the CCV reaches that in the FCV, it will not be reduced further.

3.1.3 Dominated Individuals

After we have considered all the non-dominated individuals for the next generation, any vacant places in the next generation will be filled up by dominated individuals. To decide which individuals survive in the population, all dominated individuals in the pool of all parents and offspring are sorted by niche count. The best of these individuals survive into the next generation.

3.2 Other Algorithm Details

Because of different ranges allowed, angle mutation is performed differently from radius mutation. Cauchy mutation is used in both cases [7, 8]. The scaling factor η [7, 8] in the mutation operators is fixed, but different for angle and radius. When a radius is mutated, the mutation is ‘reflected’ from the edges of the search space (e.g., if a pole currently has a radius of 0.9, and the mutation is +0.3, it will end up being $(1.0 - 0.2) = 0.8$). When angle is mutated its value is simply ‘wrapped around’ at $\pm\pi$.

Genotypes in our work consist of pairs of real numbers. Each pair describes a point in the complex plane. The Euclidean distance between matching points of two genotypes can therefore be used as a measure of distance between them. Poles and zeroes are not sorted. If two individuals happen to have the same poles or zeroes, but in different order within the genotypes, they will have a large distance between them. The distance is accumulated over all pairs and gives the total distance between two genotypes. Apart from fitness sharing, the distance calculation is also required for clustering.

Fitness sharing modifies the raw fitness of an individual according to the number of other individuals in the population which occupy the same ‘niche’. Since distance calculation is computationally expensive, each individual is only compared to a small random sample of individuals in the population. This is somewhat similar to implicit fitness sharing [9]. If the distance between two individuals as calculated above is below a threshold distance (i.e., the niche radius), the ‘niche count’ of the individual is increased by a value inverse proportional to the distance. The niche count is normalised by dividing it by the number of samples this individual has been compared with.

In our sharing algorithm, only the overall combined fitness is subject to sharing. Because fitness has been defined such that lower fitness is better, fitness sharing needs to increase the fitness value for individuals with non-zero niche counts. The shared fitness is calculated as $fitness * (1.0 + nicheCount * c_s)$, where c_s is a parameter that adjusts the degree of sharing. There are other parameters in our algorithm, e.g., the niche radius and the number of samples drawn from the population.

Fitness evaluation is a challenging issue in design, because a design task is usually multi-objective and because it is sometimes difficult to quantify the quality of a design.

First, the genotype is converted into a phenotype. This phenotype is the transfer function of the filter. The frequency response is derived by sampling the transfer function with $z = e^{j\omega}$, for $0 \leq \omega \leq 2\pi$. From the frequency response, the fitness values are computed.

For a multi-objective optimisation problem, there is seldom a single best individual that is better than all other individuals according to all objectives. In other words, there is no single objective (to be more precise, its fitness value) that can guide the evolution towards the right direction throughout the evolutionary search. We need to take all objectives into account. In our work, we have adopted a combined fitness value, computed as a weighted sum of different fitness values.

Parent selection has a large influence on the performance of EAs. The experiments reported here have been created using tournament selection with dominance using the niche count: if one individual is non-dominated, but not the other, the non-dominated individual wins the tournament. Otherwise, the individual with the smallest niche count wins. This method seems to provide the best tradeoff between the quality of the best individual in a population and the diversity in the population.

4 Experimental Results

The results produced by our evolutionary system will be compared against those generated by the human expert [3] in order to evaluate the quality of evolved filter designs and gauge the strength and weakness of our evolutionary system. In all the results given below, 300 samples have been used in the passband and 200 in the stopband. To conduct a fair comparison, fitness values have been computed for the designs given by the human expert [3] using exactly the same sampling and fitness computation methods as those used in our evolutionary system. Because of sampling and rounding, the computed fitness values for the filters are similar to but not exactly the same as those reported by the human expert [3].

4.1 Two Test Problems

Both test problems are lowpass filters [3] with slightly different numbers of poles and zeroes, cutoff frequencies, and goals for delays and amplitude.

Problem Case 1: $\omega_p = 0.2$, $\omega_a = 0.28$, maximum amplitude deviation 0.1dB, minimum stopband attenuation 43dB, group delay = 11 samples with maximum deviation 0.35, order 15 with 7 zero pairs, 1 single zero, 2 pole pairs, 1 pole single, 10 poles at the origin.

Problem Case 2: $\omega_p = 0.25$, $\omega_a = 0.3$, maximum amplitude deviation 0.3dB, minimum stopband attenuation 32dB, group delay = 9 samples with maximum deviation 0.5, order 12 with 6 zero pairs, no single zeros, 5 pole pairs, 1 pole single, 1 poles at the origin.

Genotypes representing individuals require 12 pairs of numbers for case 1 and 13 pairs of numbers for case 2. When computing the fitness for the human design in case 1, it was noted that the amplitude curve seemed to be slightly too high. When the value for b_0 was modified from -0.00046047 as given in the paper [3] to -0.000456475, the fitness value becomes very similar to that given in the paper [3]. We think this is caused either by a type in the published paper [3], or the result of rounding errors in the fitness calculations in either the paper or our implementation. We will use the corrected value in all our performance comparisons in this report.

4.2 Performance of the Evolutionary System

Table 2 shows the results of three runs for design case 1. It lists the performance of the individual with the best combined fitness, for all clusters used in run 1 and for the three best clusters in runs 2 and 3. Table 3 shows similar results for design case 2. The first row in the tables indicates the performance of the design created using the conventional approach [3].

Several observations can be made immediately from Table 2. Firstly, three out of 20 clusters for run 1 are empty, which indicate that our proposed techniques for tightening constraints and concentrating on more promising clusters appear to work.

Secondly, at least one better design that outperforms the filter designed by the human expert has been evolved in each run. According to the combined fitness that considers all design criteria, the best individuals in the 4th and 14th clusters in run 1 have achieved a performance of 6.085 and 6.046 respectively, which are better than 6.293 obtained by the human expert [3]. The best individuals in the 4th cluster in run 2 and the 2nd cluster in run 3 have achieved 5.702 and 5.760, respectively, which represent 9.4% and 8.5% performance improvement over the human design. The best individual in the 19th cluster in run 3 also outperforms the human design although only marginally.

Thirdly, the evolved designs are quite different from each other if we examine the results closely. For example, the best individual in the 4th cluster in run 2 has an extremely small maximum delay deviation in the passband (i.e., 0.043) while the individual in the 2nd cluster in run 3 has a very small maximum amplitude deviation in the passband although both designs have similar combined fitness values. These two examples illustrate the success of the clustering process.

Fourthly, there is only one evolved design (the best individual in the 9th cluster in run 1) in Table 2 that achieved a better performance than the human design according to its stopband performance, although the combined fitness is worse. This appears to indicate that the stopband performance of 0.023 in the human design is quite hard to beat. At the same time, this also points out the direction for improving our evolutionary system.

<i>Run</i>	<i>Cluster</i>	<i>PBmaxAmp</i>	<i>PBmaxDel</i>	<i>SBmaxAmp</i>	<i>combined</i>
Human Design		0.103	0.293	0.023	6.293
1	1	0.310	0.186	0.041	9.043
1	2	0.199	0.418	0.029	9.090
1	3	0.098	0.412	0.032	8.391
1	4	0.217	0.057	0.033	6.085
1	5	0.248	0.247	0.032	8.120
1	6	-	-	-	-
1	7	0.253	0.506	0.026	10.221
1	8	0.354	0.188	0.0351	8.941
1	9	0.234	0.356	0.022	8.068
1	10	-	-	-	-
1	11	0.187	1.081	0.031	15.803
1	12	0.483	1.146	0.037	20.001
1	13	0.074	0.481	0.037	9.247
1	14	0.212	0.082	0.031	6.046
1	15	0.042	0.529	0.033	8.974
1	16	0.084	0.698	0.031	11.001
1	17	0.272	0.340	0.028	8.906
1	18	-	-	-	-
1	19	0.422	0.299	0.029	10.090
1	20	0.297	0.193	0.028	7.670
2	4	0.227	0.043	0.030	5.702
2	12	0.303	0.202	0.031	8.154
2	18	0.093	0.072	0.051	6.846
3	2	0.052	0.160	0.036	5.760
3	11	0.151	0.128	0.041	6.912
3	19	0.081	0.241	0.031	6.288

Table 2: Evolved results from three runs for problem case 1, where *PBmaxAmp* indicates the passband maximum amplitude deviation (in dB), *PBmaxDel* indicates the passband maximum delay deviation (in samples), and *SBmaxAmp* represents the inverse of maximum amplitude in stopband (in dB).

<i>Run</i>	<i>Cluster</i>	<i>PBmaxAmp</i>	<i>PBmaxDel</i>	<i>SBmaxAmp</i>	<i>combined</i>
Human Design		0.271	0.437	0.030	10.099
4	1	0.398	0.262	0.042	10.783
4	2	0.486	0.334	0.044	12.526
4	3	0.219	0.651	0.044	13.006
4	4	0.268	0.361	0.042	10.467
4	5	0.638	0.256	0.047	13.657
4	6	0.948	1.650	0.065	32.524
4	7	0.707	1.429	0.034	24.656
4	8	0.512	1.226	0.066	23.978
4	9	0.781	0.326	0.032	14.320
4	10	0.514	0.626	0.071	18.555
4	11	0.614	1.234	0.049	23.453
4	12	0.407	1.174	0.045	20.349
4	14	0.544	0.809	0.025	16.051
4	14	0.190	0.564	0.041	11.633
4	15	0.317	0.319	0.034	9.764
4	16	0.271	0.454	0.049	12.183
4	17	0.519	0.267	0.059	13.807
4	18	0.864	0.287	0.034	14.913
4	19	0.236	0.731	0.048	14.438
4	20	0.294	0.334	0.038	10.152
5	7	0.404	0.144	0.047	10.190
5	14	0.426	0.147	0.044	10.156
5	17	0.359	0.262	0.038	10.016

Table 3: Evolved results from two runs for problem case 2, where *PBmaxAmp* indicates the passband maximum amplitude deviation (in dB), *PBmaxDel* indicates the passband maximum delay deviation (in samples), and *SBmaxAmp* represents the inverse of maximum amplitude in stopband (in dB).

Table 3 shows similar points as those indicated by Table 2. Both runs 4 and 5 produce a better individual than the human design, the best individuals in the 15th cluster in run 4 and the 17th cluster in run 5.

Runs 1, 2, 4 and 5 were run for 50,000 generations using a population size of 1400. A maximum of 70 individuals were allowed in the Pareto front in each cluster, and a minimum of 50 after shrinking the Pareto front. Run 3 used the same parameters, but was run over 67,000 generations. A run of 50,000 generations typically took up to 1.5 days on a 500MHz Pentium computer.

Figure 2 shows the evolutionary process of our system. The curves in the figures indicate the best fitness in the population. The figures show that the fitness was still improving even around 50,000th generation. It seems very likely that better designs would have been found if we had run the experiments longer. It is worth pointing out that it is not always true that the longer the computation time the better the solution will be. An EA can make progress in its search only when there is sufficient population diversity. Because of Pareto optimisation, fitness sharing and clustering implemented in our system, we can maintain the population diversity at a high level in our evolutionary systems for a much longer time than other EAs. That is one of the primary reasons why we could expect to achieve better performance if we had run the experiments longer.

5 Conclusions

The work presented here has shown that Pareto optimization combined with clustering can be used to produce a variety of results that otherwise would be difficult to achieve. Clustering allows the process to explore different regions in the search space without direct competition, while Pareto optimization with fitness sharing allows for an efficient, multi-objective exploitation of the different search spaces. The self-adaptive constraint mechanism that we introduced allows the algorithm to exclude unacceptable designs from the search space, without restricting the search process any more than necessary.

The design results show that it is possible to create a variety of results that are competitive with a conventionally derived design, but allow the user to chose between designs with different characteristics.

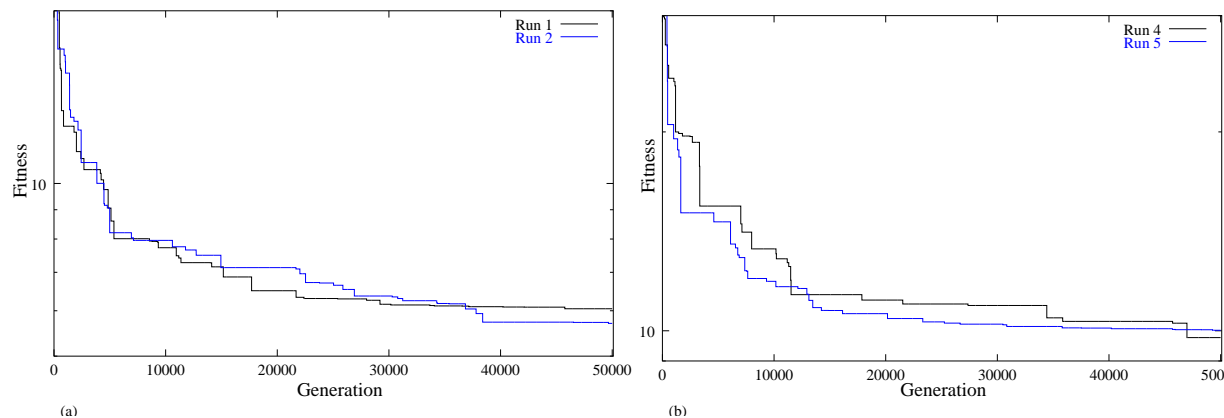


Figure 2: The best fitness in the population for (a) runs 1 and 2; (b) runs 4 and 5

ACKNOWLEDGEMENT — This research is generously supported by a grant from Marconi Communications, Ltd. The leadership and support of John Evans (from Marconi) is gratefully acknowledged.

References

- [1] X. Yao, “Following the path of evolvable hardware,” *Communications of the ACM*, vol. 42, no. 4, pp. 47–49, 1999.
- [2] X. Yao and T. Higuchi, “Promises and challenges of evolvable hardware,” *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 29, no. 1, pp. 87–97, 1999.
- [3] W. S. Lu, “Design of stable iir digital filters with equiripple passbands and peak-constrained least-squares stopband,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 11, pp. 1421–1426, 1999.
- [4] E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results (revised version),” Tech. Rep. TIK-Report 10, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, 1999.
- [5] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach,” *IEEE Transactions on Evolutionary Computation*, 2000.
- [6] Statsoft, “Cluster analysis,” 2000.
- [7] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82–102, July 1999.
- [8] X. Yao and Y. Liu, “Fast evolution strategies,” *Control and Cybernetics*, vol. 26, no. 3, pp. 467–496, 1997.
- [9] P. Darwen and X. Yao, “Every niching method has its niche: fitness sharing and implicit sharing compared,” in *Parallel Problem Solving from Nature (PPSN) IV* (H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds.), vol. 1141 of *Lecture Notes in Computer Science*, (Berlin), pp. 398–407, Springer-Verlag, 1996.