Bull, D. R., & Aladjidi, A. (1994). The optimisation of multiplier-free directed graphs: an approach using genetic algorithms. In Unknown. (Vol. 2, pp. 197 - 200). Institute of Electrical and Electronics Engineers (IEEE). 10.1109/ISCAS.1994.408938

The Optimisation of Multiplier-Free Directed Graphs: an Approach using Genetic Algorithms

David R. Bull and Alexis Aladjidi
Dept Electrical and Electronic Engineering, University of Bristol,
Queens Building, University Walk, Bristol BS8 1TR, UK.
(+44) 272 288613, Dave.Bull@uk.ac.bristol

ABSTRACT

This paper considers the problem of realising directed graphs using evolutionary optimisation methods. Graphs are constrained to have edge gains equal to powers of two and signal values at internal vertices are required to be weighted by elements of a given coefficient vector. The objective is to synthesise a graph with minimum complexity. The method is developed for the case of a single multiplicative coefficient using vertex cardinality as a measure of solution fitness and extended to the more general case of a multi-element coefficient vector with additional optimisation constraints. The potential of the approach is demonstrated using examples based on FIR digital filters.

INTRODUCTION

There has recently been increased interest in the application of genetic algorithms (GAs) [1,2] and related techniques to solving hard signal processing problems. These have included the optimisation of both fixed-function [3,4,5,6,7,8] and adaptive systems [9,10]. In the former cases however the objective has always been associated with the optimisation of quantised individual multipliers or with the satisfaction of predefined response requirements using an optimum cascade of simple filter sections.

This paper considers the problem of efficiently realising algorithms of the forms shown in (1), where either a scalar, $x[n]$ is multiplied by a fixed coefficient vector, **a,** to give an outer product, $\mathbf{y}[n]$ or an input vector, $\mathbf{x}[n]$, is multiplied by a vector, **a**, to produce an inner product, $y[n]$.

$$y[n] = \mathbf{a}.\mathbf{x}[n] \quad \text{or} \quad \mathbf{y}[n] = \mathbf{a}.\, x[n] \tag{1}$$

Such operations can be represented by a directed graph and are common to many signal processing functions including convolution and correlation. When the coefficient values are invariant, graph complexity can often be reduced by exploiting the redundancy in the coefficient-signal multiplication process, yielding a single directed graph comprising only primitive arithmetic operations (additions, subtractions and shifts).

Previous work by Bull and Horrocks [11] has used a combined heuristic-dynamic programming approach to produce a reduced complexity directed graph from a given coefficient vector. It is proposed here that GAs can also offer an efficient and flexible vehicle for primitive operator graph synthesis. To support this we present results illustrating the potential of GA based methods for the synthesis of both single and multple output multiplier-free graphs.

GRAPH SYNTHESIS

Matrix representation

A directed graph of the form considered here can be represented in matrix form (2):

$$\mathbf{v}[n] = \mathbf{F}_c^T \mathbf{v}[n] + \mathbf{B}\; x[n] \tag{2}$$

where: $\mathbf{v}[n]$ is a column vector representing the value of each vertex, $\mathbf{F}_c$ is a constant matrix containing gains for each edge of the graph and **B** is a column vector containing gains for each edge from a source vertex to each internal vertex. In addition, $\mathbf{F}_c{}^T$ must be lower triangular with zero diagonal to ensure a computable graph.

We consider first the problem of forming the product of an input signal with a single coefficient value. For example the graph of figure 1 forms the product 59x[n]. This graph can be represented in matrix form (3), where the normalised vertex values ($x[n]$=1) are in this case: $v_1[n]$=1, $v_2[n]$=17, $v_3[n]$=21 and $v_4[n]$=59. Graph vertices in this case are restricted to comprise two input adders and graph edges are constrained to powers-of-two values, thus:

$$f_c \in \left\{ k_1 2^i \pm k_2 2^j \right\} \quad \text{where } k_{1,2} \in \{0,1\} \text{ and } i, j \in \{0 \cdots 15\}$$

The goal of the optimisation process here is to obtain the target product value in the last vertex position of the vector, $\mathbf{v}[n]$ using the fewest vertices.

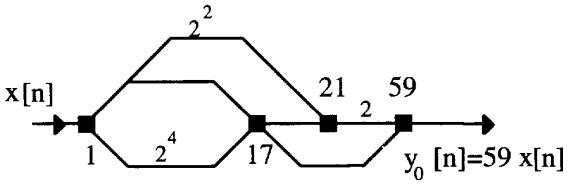

Figure 1 Graph for generating $y_0[n]$=59x[n]

$$\begin{bmatrix} v_1[n] \\ v_2[n] \\ v_3[n] \\ v_4[n] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2^4+1 & 0 & 0 & 0 \\ 2^2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \end{bmatrix} \begin{bmatrix} v_1[n] \\ v_2[n] \\ v_3[n] \\ v_4[n] \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} [x[n]] \quad (3)$$

Fitness function evaluation

The fitness of a candidate solution, F, must depend on the validity of the solution and is defined as follows:

i) if the value (v) of the last vertex does not equal the target value (v_{ref}) after repair then :

$$F = \frac{v}{v_{ref}} \text{ if } 0 \le |v| \le |v_{ref}|$$

$$\text{or } F = \frac{2v_{ref} - v}{v_{ref}} \text{ if } |v| \ge |v_{ref}|$$

$$\text{with } F = 0 \text{ if } F \le 0 \quad (4a)$$

ii) if the last vertex value equals the target value then :

$$F = 2 + \frac{n_t - n_{vertex}}{n_t} \quad (4b)$$

where n_{vertex} is the number of vertices used to compute the target value and n_t, the total number of vertices in the matrix.

Problem coding

Candidate solutions are encoded using a 2-string chromosome. The first string codes the source vertices for the input edges to each vertex, in the form of two integers representing the indices. The second string codes the edge gains as powers-of-two, encoded with 5 bits including sign. For example, the chromosome representing the graph of figure 1 would be:

string1= (1,1,1,2,2,3)

string2 = (0,0,1,0,0 ⋮ 0,0,0,0,0 ⋮ 0,0,0,1,0 ⋮ 0,0,0,0,0

=(2^4 ⋮ 1 ⋮ 2^2 ⋮ 1

:0,0,0,0,0 ⋮ 0,0,0,0,1)

⋮ 1 ⋮ 2)

The genetic operators, **crossover and mutation,** are applied to both strings. These operators are defined as usual [2], except that, in the case of the first string, mutation must be applied in such a way that any resultant number must be lower than its index position. This ensures a lower triangular matrix and hence a computable graph.

Parent selection for the reproduction phase is made via roulette wheel selection and, at each generation, the best individual is always propagated to the next generation. In addition to crossover and mutation, an operator, **Repair,** is also used which attempts to form a valid solution from a non-valid one. This operator tries to compute the target value by manipulating only those entries in the last row of the matrix but with edge gains limited to the range {-2,-1,0,1,2}. This range was found to represent a good compromise between execution time and the chance of succesful repair.

Results and statistics

100 trials were executed using a crossover probability of 0.6 and a mutation probability of 0.02 for both strings, with a range of multiplicative coefficients, for different matrix and population sizes. Performance results are summarised in fig 2 for the case of a typical coefficient value, from which it can be seen that in over 85% of cases an optimum (4 adder) solution was found. These figures have to be compared with a random search (over 100,000 individuals), which yielded a similar solution in only 0.08% of trials. An example solution found requiring 4 adders is the following:

$$\begin{cases} 33 = 2^5 \cdot 1 + 1 & 793 = 1 + 2^3 \cdot 99 \\ 99 = 33 + 2 \cdot 33 & 859 = 2 \cdot 33 + 793 \end{cases}$$

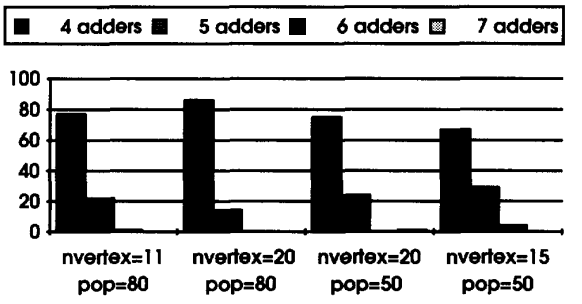

Figure 2 Results of single output graph synthesis for a coefficient value of 859.

MULTIPLE OUTPUT GRAPH SYNTHESIS

The algorithm

The procedure and representation developed in the previous section can be extended to the problem of forming a multiple-output directed graph using the minimum number of vertices. An iterative extension of the previous algorithm has been employed and is described as follows:

a) Sort coefficient magnitudes in ascending order and associate each with an internal vertex of the graph.

b) form each coefficient according to the algorithm of the section 2.

After each generation, individual solutions are propagated using roulette wheel selection as before. However, only the best (valid) individuals are retained after each target coefficient value is formed. In addition, crossover and mutation are restricted to points which do not invalidate previously formed target vertices. The function **Repair** is invoked after a specified number of generations, only if the relevant target value has not been formed.

Results

Example 1: consider the coefficient set {1,7,16,21,33}. We ran 300 trials for this example, using the same genetic parameters as used previously, with, and without, the function **Repair**. Performance results are summarised in figure 3. An example of the best solutions found with 3 adders is :

$$\{7 = 2^3.1 - 1;\ \ 16 = 2^4.1;\ \ 21 = 2^2.7 + 7;\ \ 33 = 2.16 + 1$$

Example 2: an order 31 linear-phase high-pass FIR filter impulse response with the following 8 bit coefficients {1,2,3,5,6,9,10,14,30,49,70,90,107,120,127} [11]. Results using a support matrix of 35 vertices are shown in table 1 (100 trials). An example of one of the optimum solutions with 10 adders is the following:

$$\begin{cases} 2 = 2.1; & 3 = 2 + 1; & 4 = 2.2; \\ 5 = 1 + 4; & 6 = 2.3; & 9 = 2^2.1 + 5; \\ 10 = 2.5; & 14 = 5 + 9; & 30 = 2.14 + 2; \\ 49 = 9 + 2^2.10; & 70 = 2.5 + 2.30; & 90 = 2.10 + 70; \\ 107 = 9 + 2.49; & 120 = 2^2.30; & 127 = 2.10 + 107 \end{cases}$$

Table 1:Results for the 14 coefficient example from [11] (100 trials, with repair after generation 11)

No. Add/ subs	% (pop=75, genmax=15)
10	44
11	36
12	13
13	3
missed	4

Example 3, coefficients for a video sub-band filter bank, is taken from reference [12] and has the following set of unique coefficients: {10,19,62,171,218,256,512,644,659,683,1024, 4096,4608,4703,4726,9122,9558,9728,10240}. Reducing the above set such that no coefficient is divisible by two and reordering yields the following:{5,9,19,31,109,161,171,659, 683,2363,4561,4703,4779}. A support matrix with 50 vertices was employed. An example of the best solution with 14 adders is given below:

$$\begin{cases} 5 = 2^2.1 + 1; & 9 = 2.5 - 1; \\ 19 = 1 + 2.9; & 31 = -1 + 2^5.1; \\ 109 = 2^7.1 - 19; & 161 = 2^5.5 + 1; \\ 171 = 2.5 + 161; & 659 = 19 + 2^7.5; \\ 683 = -1 + 2^2.171; & 2363 = 683 - (-1680); \\ -1680 = 2^6.1 - 2^4.109; & 4561 = -2.109 + 4779; \\ 4779 = 2^{12}.1 + 683; & 4703 = 4779 - 2^2.19 \end{cases}$$

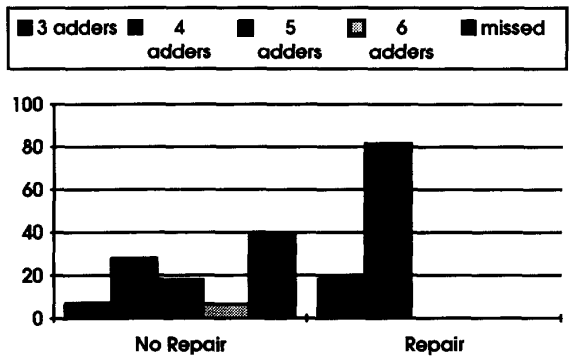

Figure 3 Results of multiple output graph synthesis showing the percentage of solutions with a given number of vertices

In terms of the number of vertices required, this last result is superior to the solution previously published in reference [12] and demonstrates the potential of the approach.

HIGHER LEVEL CONSTRAINTS

In filter implementation, for example, complexity is dictated not only by the number of graph vertices but also by precedence level relationships since these influence pipeline register requirements, and hence the area of the IC layout and latency. The fitness function described earlier has been modified in order to take in account these constraints by including an extra term, n_r, which reflects the contribution of registers in a solution.

We ran 100 trials maintaining the same genetic parameters as used previously on the test set for **example 2.** Results are given in figure 4 which shows solution distribution in terms of padding registers and adders with lines of constant fitness indicated.

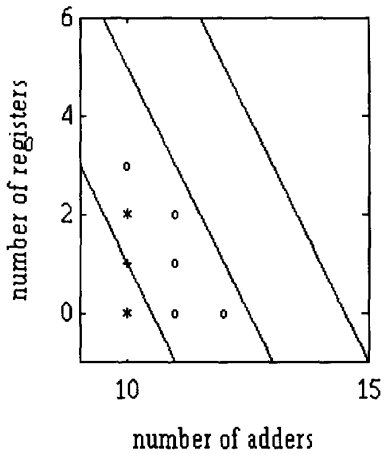

Figure 4: Results for combined optimisation of both registers and adders

UNCONSTRAINED GRAPH SYNTHESIS

The aim of this section is to demostrate the potential for GAs in synthesising a graph-based (direct form) digital filter directly from a frequency domain template. The algorithm used is a modification of that described previously, again employing two chromosomes. In this case, the signal weighting at internal graph vertices has no direct significance. Each vertex is thus represented, not by a weight, but by a vector, embodying topological information of dimension equal to the number of filter coefficients. Mutation and two point crossover are used on both strings. In addition, during reproduction and initialisation phases any solution with more than half of its coefficients equal to zero are rejected to avoid premature convergence.

The fitness function used is based solely on a measure of the error in the filter's frequency response compared with a template (the sum of the extremal errors in all stop and pass bands) and takes no account of graph complexity.

Initial trials were ran with a filter having the specifications summarised in Table 2, a lowpass filter with 3 stop bands [3]. This filter was processed using the above algorithm with the same genetic parameters as used previously. The GA was executed for 50 generations with a population of 95. Figure 5 shows an example response for a requiring 12 vertices and with coefficients {-64,0,0,96,400,128,0,-768,-1568,-2016}. This compares with an equivalent 15th order equiripple design which required 11bit coefficients to meet the specification.

Table 2: Specifications of the filter used for the trials

Band	Band edge	frequencies	δp δs
0	0.00	0.10	0.1
1	0.15	0.20	0.3
2	0.20	0.30	0.01
3	0.30	0.50	0.1

CONCLUSIONS

The results presented in this paper have demonstrated the potential of genetic algorithms as a method of synthesising directed graphs for the purpose of digital filter realisation. Ongoing work involves extensions of the unconstrained synthesis techniques to incorporate factors such as latency and pipeline register overheads into the fitness function. This should eventually facilitate the automated design of systems which are globally optimum, not just in terms of algorithmic but also in terms of technological constraints.

REFERENCES

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

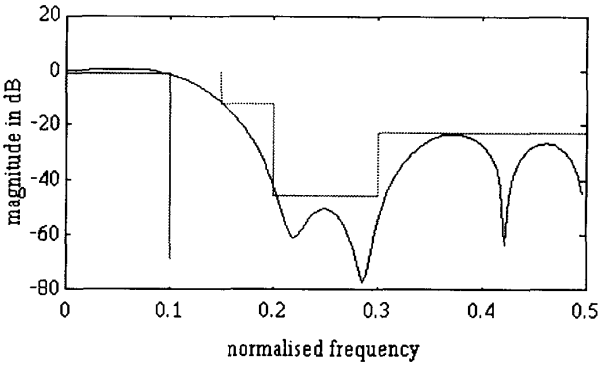

Fig 5 Frequency response of an example solution

[2] D. Beasley, D.R. Bull and R. R. Martin, "An Overview of Genetic Algorithms: Fundamentals", *University Computing,* Vol.15, 1993, pp 58-69.

[3] D. Suckley, "Genetic algorithm in the design of FIR filters", *IEE Proc. G*, Vol 138, 1991, pp 234-372.

[4] J.D. Schaffer, L.J. Eshelman, "Designing Multiplierless Digital Filters Using Genetic Algorithms", *Proc. 5th Intl. Conf. on Genetic Algorithms*, Urbana Champagne, USA, June 1993, pp 439-444.

[5] A.Roberts, G. Wade, "A structured GA for FIR filter Design", *Proc. IEE / IEEE Workshop on Natural Algorithms in Signal Processing*, Essex, UK, Nov 1993 pp.16/1-16/8.

[6] P.B. Wilson, M.D. Macleod, "Low Implementation Cost IIR Digital Filter Design Using Genetic Algorithms", ibid, pp 4/1-4/8.

[7] D. Beasley, D.R. Bull, R. Martin, DSP Algorithm Optimisation Using Expansive Coding, ibid, pp 1/1-1/10.

[8] D. Beasley, D. R. Bull and R. R. Martin, "Reducing epistasis in combinatorial problems by expansive coding"*Proc. 5th Intl. Conf. on Genetic Algorithms*, Urbana Champagne, USA, July 1993, pp.400-407.

[9] R. Nambiar, P. Mars, Adaptative IIR Filtering Using Natural Algorithms, *Proc. IEE / IEEE Workshop on Natural Algorithms in Signal Processing*, Essex, UK, Nov 1993, pp 20/1-20/10.

[10] B. S. Zhang, O. V. Patiakin, J. R. Leigh and G. Cain, "Comparison of genetic search and Darwinian adaptation in search and optimisation", ibid, pp 3/1-3/8.

[11] D.R Bull and D.H Horrocks. "Primitive operator digital filters". *IEE Proc. Part G*, Vol.138, No.3, pp401-412, 1991.

[12] D.R Bull, G Wacey, J.J Stone and J.M Soloff. "A Compound Primitive Operator Approach to the Realisation of Video Sub-Band Filter Banks", *Proc IEEE Intl. conf. on ASSP*, Minneapolis USA, May 1993, pp. 405-408.