Passive Analog Filter Design Using GP Population Control Strategies

Mariano Chouza, Claudio Rancan, Osvaldo Clua, and Ramón García-Martínez

Abstract. This paper presents the use of two different strategies for genetic programming (GP) population growth control: decreasing the computational effort by plagues and dynamic adjustment of fitness; applied to passive analog filters design based on general topologies. Obtained experimental results show that proposed strategies improve the design process performance.

1 Introduction

Conventional techniques for filter design use a particular circuit topology. The selection of a topology introduces some limitations on what designers can do, constraining the optimality of obtained results. The use of genetic programming (GP) for solving the design of this problem [5], allows exploring different component values and possible topologies. Automatically exploring different topologies by GP has competitive results compared to those ones obtained by human design. One arising problem of using GP for filter design is the code growth [13]. Some

Mariano Chouza and Ramón García-Martínez
University of Buenos Aires, School of Engineering, Intelligent Systems Laboratory
Paseo Colon 850, (C1063ACV) Buenos Aires, Argentina
e-mail: ecalot@fi.uba.ar

Claudio Rancan
Buenos Aires Institute of Technology, Master Program on Software Engineering
Av. Madero 399, (C1063ACV) Buenos Aires, Argentina
e-mail: crancan@itba.edu.ar

Claudio Rancan and Ramón García-Martínez
University of La Plata, Computer Science School, PhD Program
Calles 50 y 120, (B1900) La Plata, Buenos Aires, Argentina
e-mail: rgarciamar@fi.uba.ar

Osvaldo Clua
University of Buenos Aires, School of Engineering, Distributed Systems Laboratory
Paseo Colon 850, (C1063ACV) Buenos Aires, Argentina
e-mail: oclua@acm.org

B.-C. Chien and T.-P. Hong (Eds.): Opportunities and Challenges, SCI 214, pp. 153–158.
springerlink.com

techniques developed to solve this have been explored in restricted domains [12] but the scalability of them remains as an open issue. Passive analog filter design is a particular field of filter design domain. Conventional design techniques do not deal correctly well with the non-linear interaction of the passive analog filter component values in general topologies. In that context, this paper explores the use of some GP population control techniques applied to passive analog filter design. State of the art is presented in section 2, describing passive analog filter design state (section 2.1) and GP for filter design uses (section 2.2); problems addressed in this paper are presented in section 3; the techniques proposed to solved the addressed problems are presented in section 4; the experiment results are shown in section 5 and some conclusions and future research work are drawn in section 6.

2 State of the Art

A filter is commonly defined as a signal processing device oriented to modify the signal spectrum. This modification is generally described by the filter transfer function. This research is particularly interested in analog passive filters. The term "analog" is used because this type of filters deals with analog inputs. This type of inputs is needed in a wide set of fields; being one of the most important human-computer interaction in medicine uses. The term "passive" is used in the sense that in this type of filters only resistors, capacitors and inductors are used [9]. Classical procedures use a fixed set of transfer functions and assume a *"ladder"* circuit topology. This type of procedures is based on component value table sets where the inputs are the approximation form and order of the designed analog circuit; and the outputs are the normalized component values for those circuits. The introduction of automated design tools has decreased the use of this kind of design procedures [4, 9]. In general, classical automated design tools are software derived from the "historical" design procedures [4, 8] in which inputs are: approximation form, approximation order and topology ("ladder" is one of them); and the circuit design is obtained as an output. A contribution of the use of automated design tools is that they allow analog passive circuit sensibility analysis that may be used to detect unsatisfactory circuit designs. A disadvantage of these tools is that they do not generate new (original) topologies [5]. This research continues the exploration of the use of genetic programming [6] to analog filter design [5]. The technique consists on using an evolving procedure similar to genetic algorithms [7] instead of using conventional approaches (see section 2.1.2). A genetic programming based design procedure has as inputs: desire transfer function and a convergence metric (which compares the obtained transfer function with the desired one); and as outputs: an analog filter design (which describes topology and component values). Genetic programs may take different forms, but in general two ways are used: tree structure and lineal structure. Tree structure is used since earlier research steps in the area [6] and its philosophy is based on Lisp programming language. Linear structure philosophy is based on imperative programming languages. The commonly used structure is the tree one [1].

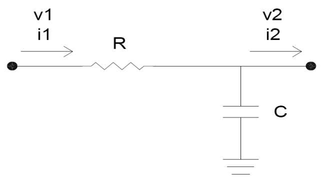

Fig. 1 Conventional representation of a low-pass filter

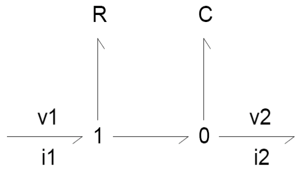

Fig. 2 Bond-graph representation of a low-pass filter

One decision in genetic programming applied to analog filter design is how the circuit is going to be represented. The current representations are: conventional [5] and bond graph based [3]. Conventional representation uses graphs where edges represent components (different type of arcs to represent different type of components) and nodes to represent the interconnection among components (edges). This type of representation takes advantage of the existence of different tools for circuit simulation *i.e.* SPICE tool [11].

3 The Approached Problems

When using genetic programming, *individual size growth* may occur [13]. This size growth becomes with two inconvenients: [a] higher memory consumption, more amount or more complex individuals implies more memory for their storage, [b] higher evaluation time, when complexity of individuals grow, the time for evaluating them grow. Both aspects limit the scalability of genetic programming because the use of computational resources (remember that part of them are used to evaluate and store individuals) cannot be used completely to improve the whole process performance. The problem of *premature convergence* occurs when the population variability decreases without having reached an acceptable solution, leaving the system stuck in a suboptimal solution. As the other mention problem it decreases the amount of computational resources applicable to improve the process performance.

4 Proposed Solution

One workaround for the identified problems may be to penalize individuals with high complexity. But this solution disturbs the development of the evolution process, because complex individuals are required to act as bridges among different solution space zones configurations. If they are eliminated the population may be stuck in several local minima. To solve these problems many strategies have been proposed [12]. We explore two of them in this paper: decreasing the computational effort by plagues, and dynamic adjustment of the fitness function. We will compare their performance solving the analog filter design problem.

Decreasing the computational effort by plagues. The excessive growth of the complexity individuals may acquire during the evolution process is called bloat.

This derives from the increasing of each generation processing time, and the amount of memory needed for the process. If the individuals with the worst fitness are eliminated best results of the evolution process are obtained during the same processing time. This improving process is called plague [2].

Dynamic adjustment of fitness function. One way of controlling individual size is to penalize them because of their size. However, this has undesirable consequences in the evolution process. Poli proposes the creation of “holes” in the fitness function to solve this problem in a dynamic way [10]. The “hole” creation process consists of the elimination of randomly selected individuals which size is over a certain number. This process put a brake to the population growth.

5 Experiments

All the comparisons use a population of 1000 individuals, 100 generations and 10 independent runs for each method. An elitist rank selection was done with an exponential probability distribution over all individuals not automatically selected in a way that the probability of selecting the individual *i* is $P(i) \approx \lambda e^{-\lambda i}$. A $\lambda = 0.002$ equal to $\lambda = 2$ over the normalized ranks in the interval $[0, 1)$. The probabilities of cross mutation and value modification (a mutation variant that only affects numerical values) was empirically optimized obtaining the values 0.1, 0.2 and 0.2 respectively. The probability of eliminating the individuals with over-average size in the dynamic adjustment of fitness function based method was fixed to 0.2. This probability value of eliminating individuals, was experimentally selected. The chosen problem for experimentation was the design of a low-pass filter with a cut frequency of 10 kHz and 1 kΩ for input and output impedance. The selected evaluation function was based on the sum of the square differences among the real and ideal transfer functions over 50 points, logarithmically distributed between 1 Hz and 100 kHz.

Relation among individuals size and evaluation time. First statistical analysis looks for establishing relation among individuals size and evaluation time. Its is shown in figure 3. It was developed over the described problem without applying any population control techniques. Fitting data with the expression $t_{eval} = A \cdot (size_{individual})^B$ $B = 2.03$ was obtained as result, indicating that the evaluation time depends on individuals size in a quadratic way (approximately).

Best individual score for each generation. The experimental results show three curves. Two of them correspond to the proposed strategies presented in this paper; the third (called reference) is related to the reference case which shows results of solving the analog filter design problem without applying any proposed strategy. Figures 4 shows minimum score average, and figure 5 shows minimum score among 10 runs using the proposed two strategies.

Relation among individuals score and time. Figure 6 shows results of minimum average score through time for both strategies, representing the computational cost in a more precise way than the results showed in figure 4.

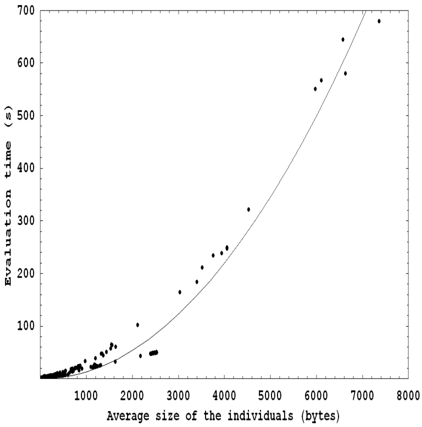

Fig. 3 Relation among individuals size and evaluation time

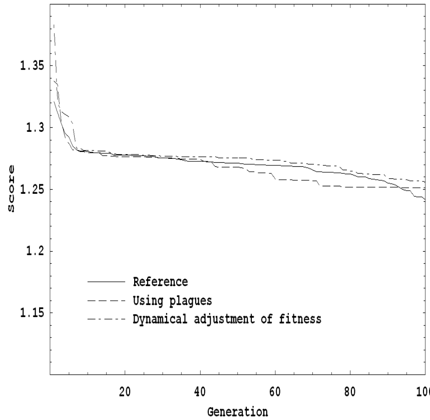

Fig. 4 Minimum score average among 10 runs

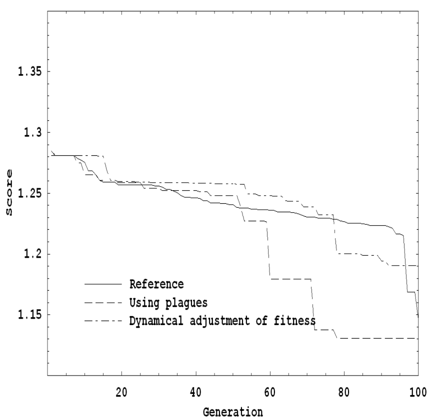

Fig. 5 Minimum score among 10 runs

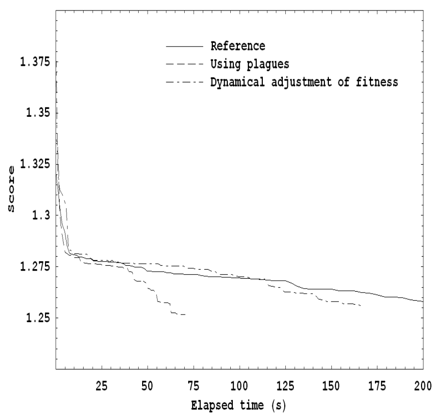

Fig. 6 Minimum average score through time

It is experimentally proven that the two proposed strategies based on GP population control strategies improve the analog filter design process performance (same solution in less time). Results shows that the strategy based on decreasing the computational effort by plagues produces better results than strategy based on dynamic adjustment of fitness function.

6 Conclusions

One limitation of the application of GP to real problems deals with that when problem complexity increases, it is necessary to increase the amount of memory to storage and the amount of time to process the population associated in order to find the solution. This paper presents a possible approach to this problem proposing the use of two different strategies for population growth control: decreasing the computational effort by plagues, and dynamic adjustment of fitness. As conventional design techniques do not deal correctly well with the non-linear interaction of the passive analog filters component values in general topologies; we select this problem to prove the two selected strategies. We found that that the strategy based on decreasing the computational effort by plagues produces better results than strategy based on dynamic adjustment of fitness function, but both proposed strategies based on GP population control improve the design process performance.

References

1. Brameier, M.: On Linear Genetic Programming. PhD Thesis. Universität Dortmund. Germany (2004)
2. Fernandez, F., Vanneschi, L., Tomassini, M.: The Effect of Plagues. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 317–326. Springer, Heidelberg (2003)
3. Hu, J., Zhong, X., Goodman, E.: Open Ended Robust Design of Analog Filters Using Genetic Programming. In: Proc. 2005 ACM Conf. & and Evolutionary Computation, pp. 1619–1626 (2005)
4. Koller, R., Wilamowski, B.: LADDER. A Microcomputer Tool for Passive Filter Design and Simulation. IEEE Transactions on Education 39(4), 478–487 (1996)
5. Koza, J., Bennett, F., Andre, D., Keane, M., Dunlap, F.: Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming. IEEE Transactions on Evolutionary Computations 1(2), 109–128 (1997)
6. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
7. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1998)
8. Nuhertz Technologies, Passive Filters Solutions (2008), `http://www.filter-solutions.com/passive.html` (accessed April 10, 2008)
9. Paarmann, L.: Design and Analysis of Analog Filters. Kluwer Academic Publishers, Dordrecht (2003)
10. Poli, R.: A Simple but Theoretically-Motivated Method to Control Bloat. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 204–217. Springer, Heidelberg (2003)
11. Quarles, T.: The SPICE3 Implementation Guide. Memorandum Nro. UCB/ERL M89/44. Electronics Research Laboratory. University of California at Berkeley (1989)
12. Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.): EuroGP 2003. LNCS, vol. 2610. Springer, Heidelberg (2003)
13. Streeter, M.: The Root Causes of Code Growth. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 443–454. Springer, Heidelberg (2003)