VARIABLE LENGTH REPRESENTATION IN EVOLUTIONARY ELECTRONICS

Ricardo S. Zebulum[1,2] **Marco Aurelio Pacheco**[2] **Marley Vellasco**[2]

1 - Centre for Computational Neurosciences and Robotics
Sussex Innovation Centre, Science Park Square,
University of Sussex, Brighton, BN1 9SB, UK
Telephone : 0044 01273 678754
Fax: 0044 01273 704499
E-mail:ricardoz@cogs.susx.ac.uk

2 - ICA - Nucleo de Inteligencia Computacional Aplicada
Pontificia Universidade Catolica,
Rio de Janeiro,RJ - Brasil
E-mail: ICA@ele.puc-rio.br

ABSTRACT

This work investigates the application of variable length representation (VLR) Evolutionary Algorithms (EAs) in the field of Evolutionary Electronics [37]. We demonstrate, through examples in both analog and digital electronics, that the varying length genotypes representation is the most natural to this particular domain of application. We propose a number of methodologies that can cope with the main issues of variable length evolutionary systems, i.e., the search for efficient ways of sampling a genome space with varying dimensionalities, the task of balancing accuracy and parsimony of the solutions and the manipulation of non-coding segments. In the latter, we emphasise the advantages of re-using genetic material to replace non-coding segments. A brief description on biological genome evolution and a mathematical formulation on the VLR genome space of analog circuits are also provided.

Key words: Evolutionary electronics, memory paradigm, non-coding segments, VLR evolutionary algorithms.

1 Introduction

Variable length representation (VLR) aims to give to artificial evolutionary systems the capability of sampling a genome space corresponding to phenotypes of different complexities. There are basically two concepts in nature that have not been adequately incorporated into artificial evolutionary systems yet: the impressive capability of complex organisms formation from simple ones[15] and the role of non-coding segments. The use of variable length data structures representing the genotypes in EAs allows the exploration of these two biological concepts.

The human genome contains about 3 x 10^9 nucleotide pairs and only about 10% to 25% are coding DNA, i.e., DNA that is transcribed to produce proteins or regulate their production[28]. DNA that does not code for genes is called non-coding DNA, or introns, and may be classified into tandem and scattered sequences. One evolutionary hypothesis says that this kind of sequence is *selfish* DNA, contributing nothing to the well-being of the organism. They may be either active or passive and they may have evolved from mechanisms such as unequal crossover and other ones. The main idea is that these facts make the genotype-phenotype mapping performed by nature far different from the one employed in traditional genetic algorithms models.

In this article, we devise a number of methodologies involving VLR evolutionary algorithms and apply them to the novel field of Evolutionary Electronics (EvolElec) [38]. This new area of study has been set up from the increasing number of EAs applications in the electronic engineering domain [24]. Evolutionary Electronics is an attractive testbed for methodologies involving variable length genotypes (VLGs), because of the large number of practical applications in this area [10]. More important is the fact that EvolElec is maybe one of the areas that can get more benefits from using these new EAs concepts. This fact can be viewed by the way a designer arrives at a particular circuit; instead of analysing possible solutions with a fixed number of components, he takes into account a large number of possible solutions with very different sizes [35]. Furthermore, parsimony is a key feature for solutions in engineering design. This way of reasoning is better mapped into EAs using variable length representation. We examine the following cases: evolution of digital circuits (comparator, multiplexers and parity detectors) and evolution of analog circuits (passive filters and bipolar transistor amplifiers).

The methodologies proposed in this work cope with the basic issues of variable length genotypes. Regarding to the problem of sampling the large genome space of varying dimensionality typical of VLR, we tried three approaches: Increasing Length Genotypes (ILG) [15], Oscillating Length Genotypes (OLG) and Uniformly Distributed Initial Population (UDIP). With respect to the problem of controlling the solution complexity, we compared two approaches: letting implicit control mechanisms actuate, by giving priority to sampling the regions of the genome space containing the smallest genotypes; and using explicit mechanisms to control the solution size, such as the Pareto Optimal concept [9]. Finally, concerning the exploration of the new possibilities arisen by the presence of non-coding material, we identified a way of re-using useful genetic material to replace non-coding loci of the chromosomes. We call this approach Evolutionary Algorithms with Memory. As we shall show, this mechanism attempts to model the advantage of diploid organisms in having genetic material preserved [28].

This work is divided into five additional sections. Section 2 introduces the field of Evolutionary Electronics and discuss the reasons why VLR evolutionary systems are a desirable approach to this field. Section 3 presents the basic methodologies developed by the authors to deal with the features of the genome space characteristic of VLR evolutionary algorithms. Section 4 presents the case studies in analog and digital electronics, as well as a discussion of the methodologies used in each case. Section 5 presents a brief discussion on the biological aspects that stimulate us to investigate this new class of EAs. Finally, section 6 concludes the work. An appendix section shows the structure of the VLR genome space for analog electronic circuits.

2 Evolutionary Electronics

Evolutionary Electronics (EvolElec) extends the concepts of Genetic Algorithms to the evolution of electronic circuits[34]. The main idea behind this field of study is that each possible electronic circuit can be represented as an individual or a chromosome of an evolutionary process, which performs standard genetic operations over the circuits, such as selection, crossover and mutation. Due to the broad scope of the area, researchers have been focusing on different problems, such as placement [8], routing [7], FPGA mapping [18], optimisation of combinational [6] [26], and sequential [26] digital circuits, synthesis of digital circuits [30], synthesis of passive and active analog circuits [20], synthesis of operational amplifiers [24] and transistor size optimisation [29]. Of great relevance are the works focusing on intrinsic hardware evolution [33] in which evolution is performed in the own silicon, exploring, therefore, all the physical properties of the medium. This particular area is called Evolvable Hardware [34].

Let us examine the question of how to deal with the problem of sizing the target circuit that is to perform a desired function. If classical genetic algorithms with fixed length genotypes are used, then a previous knowledge will be required to set the size of the target system; for instance, the number of transistors when the target design is an analog amplifier or the number of boolean gates when the target design is a digital circuit. Besides requiring a previous knowledge of the user, the other problem of this approach is that we are not using the potential of Evolutionary Algorithms to find not only the solution itself, but also its complexity.

It is therefore likely that variable length genotypes will suit better for Evolutionary Electronics [35]. In our VLG approach, we can visualise each genotype G as a collection of genes which are divided into two subsets: the subset of active genes and the subset of inactive genes. For active genes we mean those genes being considered by the EA, i.e., the coding genes. Equation 1 describes our concept of genotype.

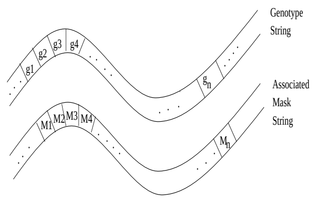

Figure 1: Genotype Model

$$G = \bigcup g_{1,2,\ldots,m} + \bigcup g_{1,2,\ldots,l} \tag{1}$$

From the above equation it can be seen that we have defined our genotype as the composition of two collection of genes, the first one containing m active genes and the second one containing l inactive genes. Our proposal is to keep the values of m and l variable along the evolutionary process, whereas keeping the total genotype length, $n = m + l$ constant. We will call the number of active genes m the effective genotype length. Each gene will encode for a piece of the electronic circuit or phenotype; this piece may be a discrete component, such as a resistor, a capacitor or a transistor, or a mathematical function performed by a set of electrical components, such as a product of boolean variables.

Our assumption is that we may employ similar VLRs methodologies to different problems in EvolElec, ranging from boolean function synthesis to the design of high gain amplifiers, since all these problems share the common feature of phenotype sizing determination.

3 VLRs Methodologies

VLR evolutionary algorithms have been investigated more intensively in the beginning of the 90's [15]. Examples of VLR evolutionary algorithms are the Messy GAs [12], LS-1 classifiers [32], genetic programming [19] and the Species Adaptation Genetic Algorithm (SAGA) [15]. A survey on this algorithms may be found in [15].

We describe in this section our VLR methodologies, which borrow some concepts from the SAGA methodology [15]. The following issues are described below: genotype model, sweeping strategies, genotype size control and use of non-coding segments.

3.1 Genotype Model

Figure 1 illustrates the genotype model employed in this work, which is based on the concept of active and inactive genes described in equation 1.

Our model adopts a genotype formed by a string, as in standard genetic algorithms. The string is a collection of genes, g_1, g_2, until g_n, each one with the same number of loci, L, that is usually greater than 1 and dependent on the application. Our model differs from standard

Genetic Algorithms basically from the fact that there is another string linked to the main one, called associated mask string. The bits of this auxiliary string work as masks for the genotype main string. A particular gene g_i, in the main string, will be active if the associated value M_i in the mask string equals 1. If M_i equals 0, then the gene g_i will be inactive. This simple mechanism is used throughout this work to provide a computational implementation for our VLR Evolutionary Algorithm. According to the representation used in equation 1:

$$\mathrm{m} = \sum_{i=1}^{n} M(i) \quad ; \quad l = n - m \tag{2}$$

i. e., the effective genotype length is given by the number of active genes. We have been careful, along our experiments, to use a value of n big enough to avoid upper limit boundary effects [1].

Besides the standard evolutionary algorithm operators, i.e., crossover and mutation, we have included another genetic operator, called $change - mask$ operator, which can either activate or de-activate a particular gene g_i, by changing the value of $M(i)$. This operator will be applied in each locus of the associated mask string with a particular probability and it is not particularly inspired in any *explicit* mechanism used in nature; rather, it is a way to provide sampling over the whole genome space consisting of phenotypes of different complexities. We should mention that the concept of genotype masks has been used in genetic algorithms as a way of preparing for uniform crossover [11]; here we employ the same concept within a different context.

In addition to the $change - mask$ operator, the $crossover$ operator will also actuate in the genotypes' masks. When two genotypes exchange genetic material through crossover, the correspondent masks of each gene will be similarly exchanged. Even though in some VLR evolutionary algorithms, such as the SAGA methodology, the crossover operator is made in such a way to maximise the similarity between the genotypes segments exchanged [15], in this case the splicing of genetic material is made freely between random individuals in random positions of the genotypes.

As a way of investigating more easily the effects of selection pressure, we have used exponential selection [4] in our applications. It will be shown that the selection pressure plays a key role in the way the varying dimensionalities of the genome space are sampled for this particular domain of applications.

3.2 Sweeping Strategies

Sweeping strategies regards to the way in which the different dimensionalities of the genome space are sampled by the EA. We devised three methodologies to handle this problem: the Increasing Length Genotypes (ILG) methodology, the Oscillating Length Genotypes (OLG) methodology [1] and the Uniformly Distributed Initial Population (UDIP) methodology.

[1]Conceived by Anil Seth, from the Centre for Computational Neurosciences and Robotics of the University of Sussex)

3.2.1 Increasing Length Genotypes (ILG)

This strategy has been devised within the context of the SAGA [15] methodology and it seems to be the most suitable for engineering problems. The Increasing Length Genotypes methodology is inspired by the biological formation of complex organisms from simple ones [28]. We initialise all the genotypes with the same mask, in which just a few bits are set to one, activating few genes; simple phenotypes are therefore produced initially. Except for the fact that we fix the number of bits set to 1 in the initial mask, its general pattern is randomly initialised. Due to the philosophy of this methodology, the $change - mask$ operator may only activate genes in this approach. As it will be shown, the population will tend to remain homogeneous, avoiding problems of mating between radically different genotypes, as described by Goldberg [11].

There are two basic advantages in using the ILG methodology: the implicit control mechanism for the solution sizes and the gain in speed.

The former advantage can be better viewed from Figure 2, that shows the step-like shape of the average fitness of the genome space in a particular evolutionary electronics application (this curve is justified in the appendix section). The ILG strategy starts sampling the points near the origin of the complexity axis, going toward more complex genotypes; however, as long as the average genotype size of the population increases, the selection pressure towards genotype growth decreases, due to the stabilisation of the average fitness curve. Therefore, it is likely that the final solutions produced by the ILG strategy will have a complexity near the value C_{opt}, which is usually close to the optimal.

In order to show how this idea expresses in practice, Figure 3 displays both the average genotype size along evolution and its correlation with the average fitness for a particular experiment using this sweeping strategy. The experiment consists in the evolution of a 6-multiplexer, in which all the population start with one gene and are allowed to grow until ten genes. It can be seen (Graph B) that the average population genotypes' size stabilises a little above the optimal value for the 6-multiplexer, 4 genes (see case studies section), instead of growing until the upper boundary; furthermore, the high correlation in the beginning and subsequent sharp fall (Graph A) shows that the selection pressure towards growth is high in the first generations and decreases along the process. This results are averaged over ten executions.

The gain in speed in the ILG strategy stems from the fact that circuits with complexity much beyond the one of the optimal solution, whose simulation is more time consuming, are not sampled.

3.2.2 Oscillating Length Genotypes Strategy (OLG)

This strategy is a variation from the ILG strategy. Like in the ILG strategy, the initial population is made up of small genotypes, but they are now allowed to decrease in size. The change-mask operator rates are switched between positive (genes being activated) and negative values (genes being deactivated). One possibility is to assign to this operator a rate obeying a sinusoidal

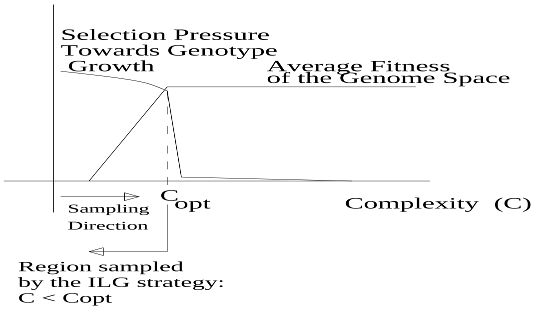

Figure 2: Increasing Length Genotypes in Evolutionary Electronics

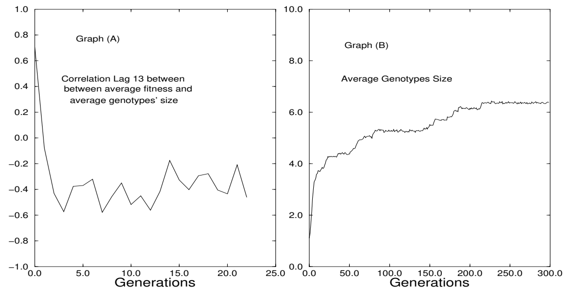

Figure 3: Correlation and genotype size in a particular EvolElec case study

function. The main purpose of this strategy is to create pathways from large to smaller genotypes with similar fitness values.

3.2.3 Uniformly Distributed Initial Population Strategy (UDIP)

Instead of starting with a population of small genotypes, the initial genotype sizes are now randomly assigned to values ranging from one to the maximum number of genes n. Therefore, the initial population is heterogeneous in terms of genotype size. The $change - mask$ operator rate is assigned to a constant and low value. The objective of this strategy is to find out the EA behaviour with heterogeneous populations.

3.3 Genotype Size Control

Genotype size control regards to the problem of balancing accuracy and parsimony of the solutions. In [40], Zhang devises a taxonomy of growth control schemes, classifying them into direct and indirect, non-adaptive and adaptive and off-line and on-line. Within the genetic programming framework, many approaches, such as adaptive parsimony pressure [39], deleting crossover [4] and explicitly defined introns [27], have been proposed to tackle the bloating phenomenon [25]. Another more general approach is the MDL methodology [17]. As we have previously stated, some sweeping strategies can give a solution size control mechanism for free. Within our EA framework and concerning our area of application, it has been verified that implicit mechanisms produce good results. However, for the sake of comparison, we propose in this work the use of the Pareto-Optimal concept for multi-objective optimisation [9] as a method to balance parsimony and correctness.

The main advantage of using Pareto is that we do not need to incorporate terms of correctness and parsimony into the fitness evaluation function, which would imply in aggregating quantities of different meanings into it. Each generation, a list of dominated and non-dominated [9] individuals, regarding correctness and parsimony, is produced by the evolutionary algorithm. After making this list and prior to the selection and reproduction process, the fitness values may be changed if the present generation falls within a particular interval period. This change consists in assigning the lowest possible fitness value to all dominated points, regardless of their actual fitness; and maintaining the fitness values of the non-dominated points. Depending on the selection methodology employed, the dominated individuals may be partially or totally eliminated from the reproduction process. The value of the period in which the fitness values are changed is usually kept around 20 generations. Reducing or increasing this period will respectively increase or reduce the selection pressure towards fit and parsimonious solutions.

Even when this control mechanism is not used, the authors included only the correctness term in the fitness evaluation function, within all the sweeping strategies defined above.

3.4 Use of Non-Coding Segments and the Memory Paradigm

Within the genetic programming framework, non-coding segments have been studied from the bloating phenomenon perspective [25], i. e., tree size explosion accompanied by a reduced performance of the optimisation process [4]. According to Banzhaf et al [1], the advantage of variable size genomes is their ability to choose their own representation, whereas the apparent disadvantage is the production of non-coding segments. They concluded that while introns do not affect the fitness of the individual directly, they may affect the likelihood that the individuals will survive, by protecting them against destructive crossover [25].

Another implicit role of non-coding segments in artificial evolutionary systems regards to the formation of neutral networks. Harvey and Thompson [14] made the first work to investigate this implicit functionality of non-coding segments in the context of Evolutionary Electronics. They have used the concept of neutral networks of fitness landscapes in the evolution of FPGA's circuits. Neutral networks are a connection of points representing different genotypes which have the same fitness. These networks are formed if the genetic encoding has a number of redundant loci (called junk DNA), which are never used to determine the phenotype. They introduced the notion of potentially useful junk DNA, which refers to loci in genotype that are functionless within a particular context, but given different values elsewhere, they may well become significant. It has been verified that neutral networks played an important role in the evolution of a two-tones recogniser circuit [33].

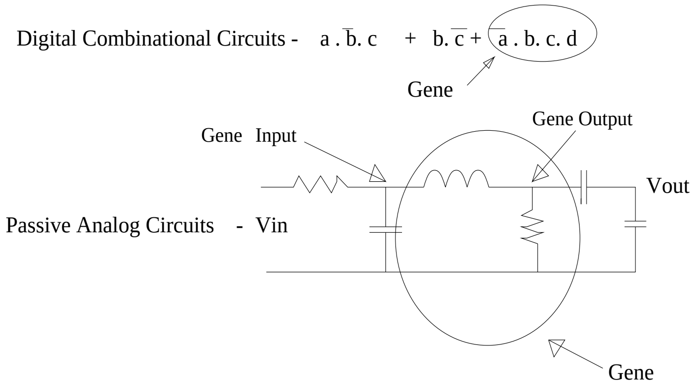

Figure 4: Memory Paradigm in Evolutionary Electronics

In this work, we make an **explicit** use of non-coding segments; instead of initialising randomly the genes' contents as they are being activated, we may pick out a gene known a priori to be probably fitter. In order to use this methodology, our representation will have to be such that the evaluation of each gene as an individual unit will have to be enabled. This assumption is plausible from a biological point of view, since each gene can roughly be said to encode for a protein, having an effect in the organism that can be assessed.

In the case of Evolutionary Electronics, this kind of representation can be achieved in both digital and analog circuits. In the former, each gene may account for a boolean product of input variables and the whole genotype will encode for the overall sum of products representing the circuit. In the latter, each gene may encode for a mesh of the circuit. This is shown in Figure 4.

This methodology requires the existence of a central data structure in the EA, which will behave as a memory; as fitter genes come up in the beginning of evolution, they are stored and then re-used as the genotypes grow in size. Evolutionary algorithms using the memory paradigm will have the following advantages over conventional ones:

1. Overcoming the short schemata processing limitation of classical GAs, since the highly fit short sized schemata, initially spread through the population, are now *explicitly* gathered to constitute longer sequences of genes;

2. Saving useful genetic material from being lost once they appear in low fit genotypes, because they are now stored in the memory; this is akin to the diploidy effect of preserving genetic material in natural evolution.

The main consequence of the use of the memory paradigm and its associated representation is the departure from the implicit schemata processing concept of traditional genetic algorithms, in the case of building blocks larger than the gene size; though, the implicit schemata theorem still holds at the gene level.

Another advantage of this representation is the possibility of carving up the genome space. To each gene a unique index given by:

$$index_{1,2,\ldots,K^n} = \sum_{i=0}^{n} L_i\, K^i \tag{3}$$

may be assigned. L_i is the value of the locus i of the gene, that has total of n loci, and K is the size of the representation alphabet (assuming binary or integer representation). By doing this, a huge genome space can be uniformly decomposed in many slices according to the genes' assignments. If we have a total of S slices, each parallel running evolutionary algorithm $EA_j (1 \leq j \leq S)$ will sample the following region R_j:

$$\frac{K^n}{S}(j-1) \leq R_j \leq \frac{K^n}{S} j \tag{4}$$

A simple penalty term is applied to each EA_j fitness evaluation function, so as to induce them to sample their respectives slices of the genome space.

4 Case Studies

This section shows the results on the application of the above concepts in two major areas of electronics, digital and analog. Throughout the section, we show our results using a set of performance statistics that takes into account fitness and parsimony, given by:

$$< r > = \frac{U_T(f). < F >}{< G >} \tag{5}$$

$$r_{best} = \frac{U_T(f).F_{best}}{G_{best}} \tag{6}$$

where $< F >$ and F_{best} are the final average and best fitness respectively over a set of executions, and $< G >$ and G_{best} are the final average genotypes' size and best individual genotype size over the same set of executions. $U_T(f)$ is a unit step function over a fitness threshold T selected by the user. If the achieved fitness is better than the specified threshold, the ratios defined above will be greater than 0, so that the higher their values, the better the methodology.

4.1 Digital Circuits

We examine here the application of VLR EAs in the synthesis of combinational digital circuits. Given a truth table or a list of minterms for a particular boolean function, the task is to find a minimum digital circuit that implements this function [19]. There are two main reasons for studying combinational circuits: first, they are widely used in computer systems; second, the evolution of combinational functions has been classically used as a testbed for evolutionary methodologies [22]. Evolutionary algorithms may be used in combinational function optimisation together with

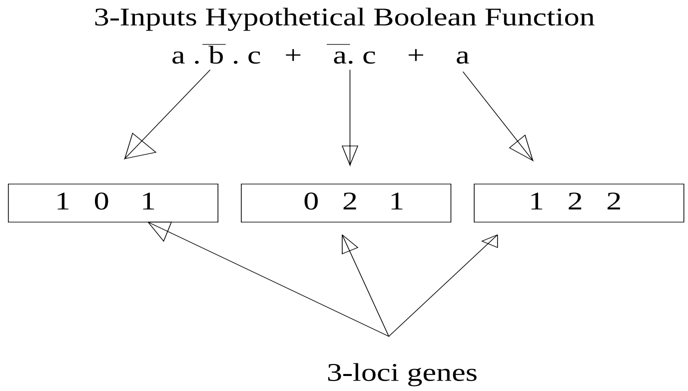

Figure 5: Boolean functions representation

simplification heuristics, providing results that are competitive with the state of art in CAD (Computer Aided Design) [6][26]. Here we handle the problem of boolean function synthesis without any heuristics, which turns out to be a very hard and interesting problem.

4.1.1 Problem Modelling

Considering that any boolean function can be represented as a sum of product terms, we have represented the genotypes as a collection of genes that encode for any possible product of boolean variables, complete or incomplete. For a $n-input$ digital circuit, each gene will have n loci, which can take three possible values: 0 if the correspondent input variable is complemented, 1 if the input variable is uncomplemented and 2 if the input variable is absent. This is shown in Figure 5. We will call, in this work, the order of boolean term as the number of "$2s$" in its correspondent gene, so that higher order genes are the more parsimonious ones (In a previous work [36], the authors used the so-called gate level representation to combinational circuits, but the poor results stimulated us to change to the present representation). The use of variable length representation in this problem is a natural choice, since one does not know, in principle, of how many terms the minimal solution is made up of, i.e., the degree of simplification of the boolean function.

The number of fitness cases for combinational circuits increases exponentially with the number of inputs and, as a consequence, the hardness of the problem increases non-linearly as well. The authors have identified that, in a general way, as the number of inputs n surpasses the value of 6 or 7, special methodologies are required, such as inserting memory in the evolutionary system or carving up the genome space. Nonetheless, as it will be shown, the structure of the target Boolean function also plays a major role in the determination of the problem complexity.

The fitness evaluation function takes only correctness into account, by measuring the number of hits according to the truth table; additionally, we also included a term that strongly penalises genotypes with repeated genes and genes that produce '1' outputs where the truth table is '0', as this behaviour is the unique that can badly affect the whole genotype performance.

4.1.2 Applications

We show results for some basic boolean functions, such as multiplexers, comparators and parity functions.

The class of multiplexer combinational functions is one in which the ratio Number of terms of the optimal solution/Number of Minterms is minimal, i. e., this combinational function allows maximum simplification. We examine in this section the evolution of 6 and 11 input multiplexers.

The 6-input multiplexer is a standard combinational circuit commonly used for the evaluation of evolutionary methodologies [19][30]. This combinational function consists of 32 minterms, which are simplified to 4 terms that make up the minimal solution. According to our representation, there are $3^6 = 729$ different genes; the task of the Evolutionary Algorithm is to find a genotype in which only the four genes representing the minimal terms are active. There are a total of ${C_{729}}^4 \cong 7.00x10^{10}$ possible combinations of four genes in this genome space. Ruling out the task of evolving also the solution size, this number provides a good measure of the problem complexity.

The results are summarised in table 1. It presents, for each methodology, the percentage of success, the average complexity and their ratio r, as a measure of performance. All the experiments have been carried out using one-point crossover, with a rate of 99%, a mutation rate of 5% per genotype locus, population of 30 individuals and 100 executions. Each execution lasts less then a minute in a Sparc Sun station model Ultra 2. Each individual can have a maximum of 20 active genes, corresponding to a genotype of 20 x 6 = 120 loci. Moreover, in this table we have also included some results obtained in [4], in which explicit mechanisms to control the genotype size were used within the genetic programming methodology. Both our VLR methodology and genetic programming process a similar number of individuals per execution in this comparison (9,000 in our methodology and 10,000 in the GP methodology). The following approaches are being compared [2] :

1. EA (1) is the evolutionary algorithm employing the ILG strategy;
2. EA (2) is the evolutionary algorithm employing the UDIP strategy;
3. EA (3) is the evolutionary algorithm employing the OLG strategy;
4. EA (4) is the evolutionary algorithm employing the ILG strategy with Pareto control;
5. EA (5) is the evolutionary algorithm employing the UDIP strategy with Pareto control;
6. EA (6) is the evolutionary algorithm employing the OLG strategy with Pareto control;
7. GP (1) is the genetic programming methodology using deleting crossover with parsimonious pressure [4];

[2] To compare the complexity of the solutions, we estimated the number of gates necessary to implement the boolean terms obtained in our solutions, and assumed that each GP tree node was equivalent to a gate

Table 1: Survey on the 6-Multiplexer Evolution

Methodology	Psucc(%)	Average Complexity(gates)	r
EA (1)	88	10.64	8.27
EA (2)	84	12.78	6.57
EA (3)	92	12.86	7.15
EA (4)	93	11.96	7.78
EA (5)	91	13.10	6.95
EA (6)	92	12.42	7.41
GP (1)	57	12.00	4.75
GP (2)	62	57.73	1.07

8. GP (2) is the genetic programming methodology using deleting crossover without parsimonious pressure [4].

From this table, one can conclude first that the evolutionary algorithm approaches developed by the authors outperform the genetic programming approaches, both in terms of performance and complexity of the solutions. Second, we can see that the three VLGs methodologies tend to have a very similar performance, either with or without Pareto control. Using the ratio $r = Percentage\ of\ Success/complexity$ as a measure of the performance of the evolutionary systems, we can see that the ILG methodology slightly outperforms the other ones. It can also be observed from this set of experiments that, when the Pareto technique is used, the percentage of success increased slightly for EA(1) and EA(2).

The graphs of Figure 6 show the average behaviour of the population for the three strategies employed. Graphs (A) and (B) show the average genotype length and variance of the genotype length respectively for the ILG strategy; graphs (C) and (D) show the average genotype length and variance of the genotype length respectively for the OLG strategy; and graphs (E) and (F) show the average genotype length and variance of the genotype length respectively for the UDIP strategy.

We can observe that, in the ILG and OLG strategies, the average size of the genotypes starts at the value of one gene and stabilises with four genes, which corresponds to the optimum number of boolean terms for the 6-multiplexer. The variance of the genotypes' size along the evolutionary process is usually low and the population stays homogeneous through evolution. On the other hand, in the UDIP strategy, the initial average genotypes' size is around 10 (Graph E), reflecting the fact that the initial individuals have their sizes randomly assigned between 1 and 20 genes. In the first generation, due to the high selection pressure, there is a sharp fall in the average size of the genotypes and also in the variance of the population (the initial values are not clear in Graphs E and F, 9.3 and 3.8 respectively). This is due to the fact that, in the beginning, smaller

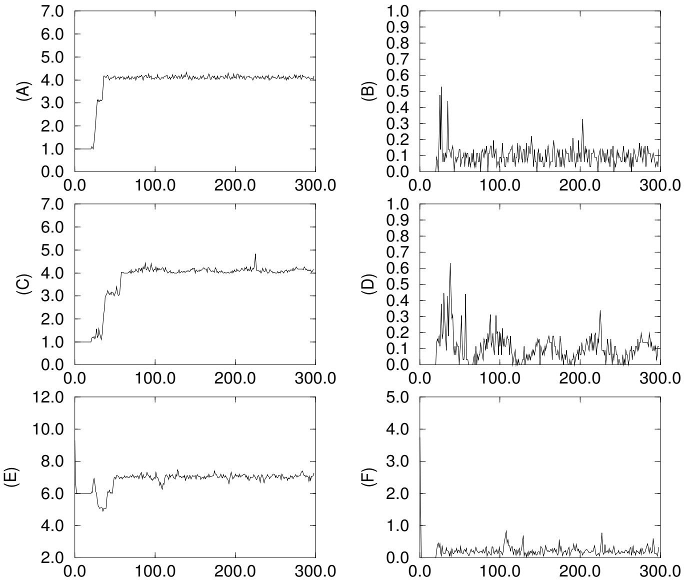

Figure 6: 6-multiplexer evolution using ILG (A, B), OLG (C, D) and UDIP (E, F) strategies (Axis X is the number of generations)

individuals usually have higher fitness. It is interesting to notice that the low variance in the genotypes' length is one of the assumptions of the Species Adaptation Genetic Algorithm [15].

In order to verify the effect of the selection pressure, we ran two extreme cases for the 6-multiplexer: exponential selection with very low selection pressure (parameter $c = 0.95$ [4]) and exponential selection with very high selection pressure (parameter $c = 0.4$). The results, in terms of average genotypes' length, using the OLG methodology, are shown in the graph of Figure 7. It can be seen that, for high selection pressure, the average genotype length reaches an almost constant value quickly, whilst it tends to oscillate when lowering the selection pressure. It is also shown, in this graph, the change-mask operator rate (in a different scale), given by a slowly decaying sine wave.

The evolution of larger boolean functions, such as the 11-multiplexer, 8-comparator and the 8-parity, can be accomplished by the introduction of enhancements in our basic methodology, such as the *partition of the genome space* and the use of the *memory paradigm*. In the former, we apply equations 3 and 4, for $K = 3$; the latter consists in storing all the genes that do not produce a '1' output when the correct output is '0'.

The 11-multiplexer is a combinational function defined by 1,024 minterms that are simplified to only 8 terms. There are $3^{11} = 177,147$ different genes according to our representation. The complexity of picking up the 8 minimal terms from the total number of genes can be written as ${C_{177,147}}^{8} \cong 2.41x10^{37}$, 10^{27} times higher than the complexity of the 6-multiplexer. This statistics itself precludes the use of our previous EAs without any enhancement. We have carved

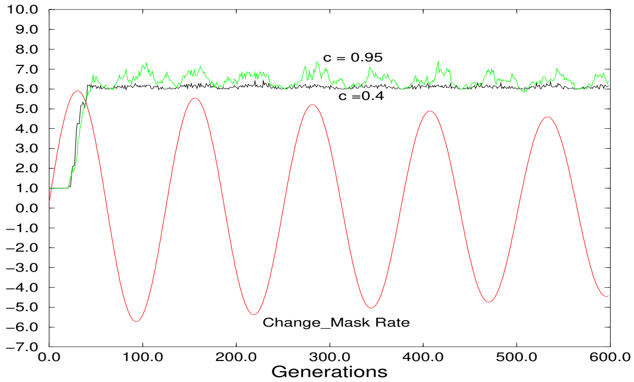

Figure 7: Effect of extreme degrees of selection pressure in the average genotype length for the 6-multiplexer using the OLG methodology

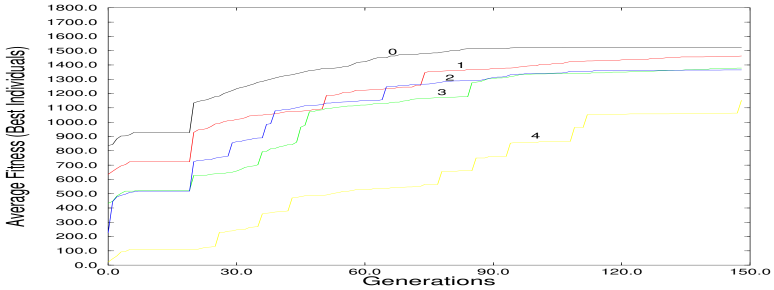

Figure 8: Average fitness in the 11-multiplexer experiment (EA number assigned to each curve)

up **uniformly** the genome space into five slices, using thus five parallel EAs and the memory paradigm. The genes found by each parallel EA may be composed in the overall solution, since they account for non-superposing regions on the genome space and there is no epistatic effect in our representation. Using this criteria, we have obtained **six** solutions over ten executions of our EA; **four** of them included the eight optimal genes. The results compare very well with other references [19][30], taking into account the fact that we did not use any specific knowledge of the multiplexer functions in the design of the evolutionary algorithm. Furthermore, our methodology promises to be less time consuming then other methodologies applied to the problem; each run processes a total of 22,500 individuals (30 individuals, 5 EAs and 150 generations), against 36,000 (solution found in 9 generations over a population of 4,000 individuals) in [19]. Figure 8 shows the average fitness of the five EAs over ten executions; the best possible fitness is $2^{11} = 2{,}048$, achieved by neither processes, since the solution is now distributed.

The effectiveness of the re-use of genetic material when initialising non-coding segments along evolution may be demonstrated by the evolution of the 8 bits comparator. This device implements

Table 2: Survey on the 8-comparator evolution

Methodology	Psucc(%)	$< G >$	$< F >$	$< r >$
OLG with Memory	100	22.1	256 over 256	12.19
ILG with Memory	80	18.5	255.8 over 256	13.83
OLG without Memory	0	8.1	243.8 over 256	0

the logic function $A > B$, where A and B are 4-bit numbers, resulting in a total of eight inputs. This combinational function is specified by 128 minterms, which may be reduced to 15 terms, using the standard CAD tool Espresso [5]. Whereas the solution of the multiplexer functions are made up of high order terms (highly simplified terms) and the ones for the parity functions are made up of the lowest order terms (the own minterms), the solution to the 8-bit comparator includes both high order terms and minterms. There are a total of $3^8 = 6561$ possible genes. The complexity of gathering 15 terms can be calculated by ${C_{6561}}^{15} \cong 10^{45}$. Table 2 shows the average performance of our methodology, when using the OLG strategy and the ILG strategy, both with the memory paradigm, and the OLG strategy without the memory paradigm. The results are expressed in terms of the performance statistics defined in equation 5; we chose as the minimum acceptable fitness threshold T the value of 250 over 256 hits, so that the last methodology in the table has $< r > = 0$. From this table, it can be seen that the re-use of genetic material in non-coding regions produces an extraordinary benefit to the performance in this case. It can also be verified that the ILG methodology arrives at more parsimonious solutions in terms of number of genes; the best solution achieved had a total of 16 genes, one gene more than the optimal. The methodologies employing the memory paradigm took around four hours to run 10 executions in a SPARC classic workstation.

Regarding to the 8-parity combinational function, it is specified by 128 non-simplifiable minterms, leading to a complexity of ${C_{6561}}^{128} \cong 10^{271}$. The genome space has been divided into ten regions, and one EA has been used to sample each region. Figure 9 shows the best individual fitness along evolution for each running evolutionary algorithm; the ones that sample regions with more minterms reach higher fitness values. In the end, all 128 minterms have been found.

4.2 Analog Circuits

The area of analog circuits design can get more benefits from the use of artificial evolutionary systems than the area of digital design, since in the former there are not, so far, efficient CAD tools available [23]. We present results on passive filters and transistor based amplifiers evolution.

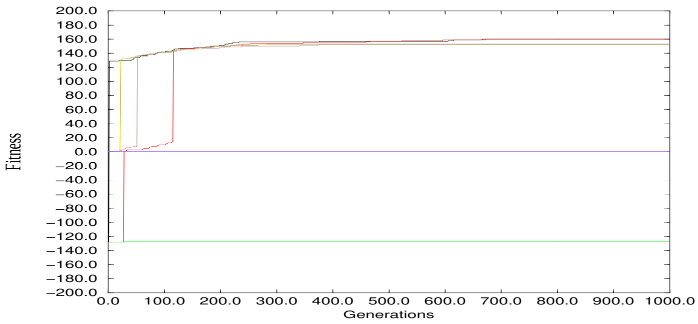

Figure 9: Fitness of the best genotype along the evolutionary process for each of the ten running EAs of the 8-parity problem (1 execution)

4.2.1 Passive Filters

The use of artificial evolutionary systems in the design of passive filters has already been studied before through Genetic Algorithms [13] and Genetic Programming [21]. Our approach goes beyond the theoretical purposes of applying VLRs in filter design. It helps to evaluate its practical usage, since the EA uses only standard electric components [16]. This section extends the results shown in [37].

Each gene encodes a particular electric component, by choosing its nature (resistor, capacitor or inductor), its value, from a set of preferred values [16] and its connecting points. Any arrangement of components is allowed, which makes possible the exploration of new topologies. Based on Koza's previous works [20][21], we have devised the following fitness evaluation function:

$$\text{fitness} = \sum_{i=1}^{n} w_i |Target_i - Output_i| \tag{7}$$

According to equation 7, the fitness is the weighted sum of the deviations between the frequency response specification, $Target_i$ and the frequency response obtained $Output_i$. The weight w_i takes a maximum value for frequency points inside the passband, an intermediate value for frequency points inside the stop band and a minimum value for other frequency points. The weights w_i assume the values of 20, 10 and 1 for these three cases respectively. The fitness value defined above is further manipulated so as to assign to fully compliant circuits (those with fitness equal to zero in equation 7), a maximum defined score of 1,000 points. Although this fitness equation worked well, setting the values of the weights has been an interactive and time consuming process. To evaluate the circuits, we have used the SMASH simulator, from Dolphin corporation [31].

In order to demonstrate the flexibility of our methodology, we present results for the design of low-pass, band-pass and notch filters. In all cases we have used VLRs methodologies with genotypes made up of 20 genes, exponential selection and one point crossover.

Table 3: Results on Analog Filters Evolution - LP = low-pass, BP = band-pass; UDIP and OLG strategy employs the Pareto control strategies

Methodology	Statistics	LP	BP	Notch
ILG	$< r >$	82.5	98.1	57.4
ILG	r_{best}	70.1	66.3	97.6
UDIP	$< r >$	73.8	64.8	53.3
UDIP	r_{best}	69.8	59.3	54.8
OLG	$< r >$	82.5	0	0
OLG	r_{best}	81.3	0	0

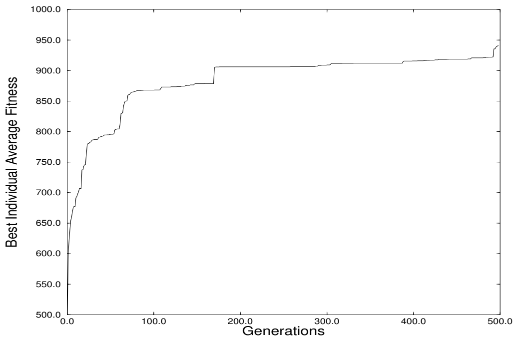

Figure 10: Best genotypes average fitness for the low-pass filter

Table 3 summarises our results in terms of the performance statistics defined in equation 5 and 6; we defined the minimum acceptable fitness threshold to be of 800 over 1,000. The results shown in this table refers to five executions of 500 generations with a population of 40 individuals. Each execution takes around two hours in a SPARC station model Ultra 2. It can be seen that the ILG strategy outperforms the other ones for all filters, except for the low-pass filter, in which the OLG strategy outperformed the others. Curiously, the OLG strategy failed to achieve the minimum threshold fitness for harder filter specifications, i.e., band-pass and notch.

Figure 10 shows the average fitness of the best genotypes over the executions for the low-pass filter experiments using the ILG strategy. The filter is specified to have a passing band below 1,000 Hz and a stop band above 2,000 Hz [21]. The best circuit achieved a fitness of 981 over 1,000. The graphs of Figure 11 show the average genotype length along evolution for three over five executions, for all strategies. From this graphs we can see that, whilst the ILG strategy produces a constant trend towards growth, the OLG strategy shows more tendency to oscillation and the UDIP strategy shows just a slight tendency towards genotypes' growth. While the variance of the genotypes' size stay at low value for the ILG and UDIP strategy (not shown here), the OLG strategy exhibits peaks along evolution, as shown in Figure 12.

The band-pass filter has been specified to accept frequencies between 2,000 and 3,000 Hz, and

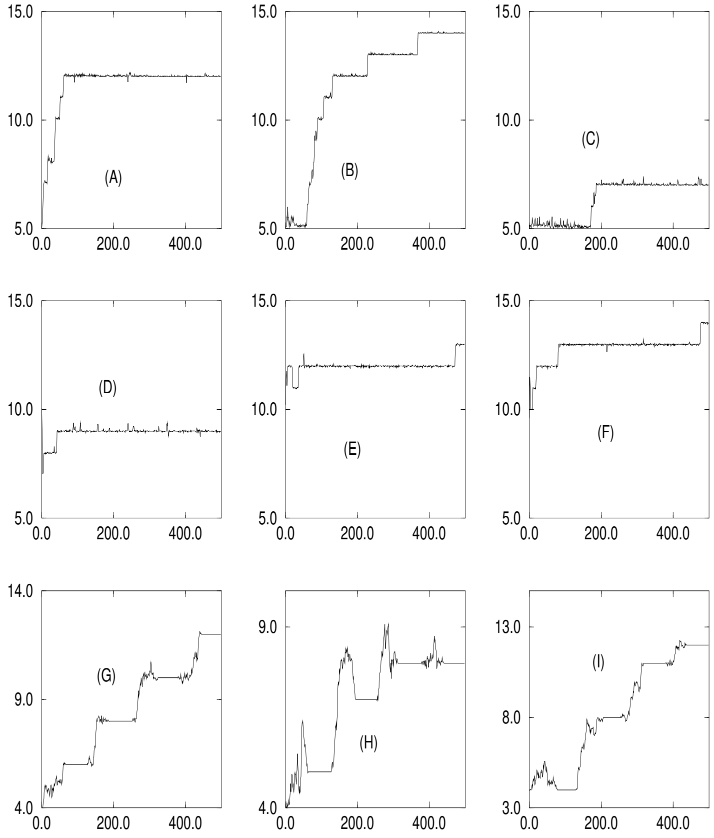

Figure 11: Average genotypes' sizes for the low-pass filter using ILG (A, B, C), UDIP (D, E, F) and OLG (G, H, I) strategies (Generations x Number of Genes)

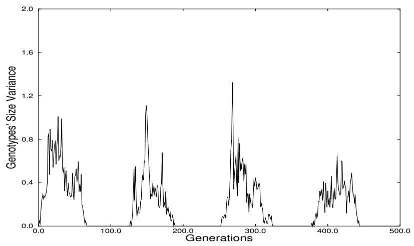

Figure 12: Genotypes' size variance for the low-pass filter using the OLG strategy in a particular execution of the EA

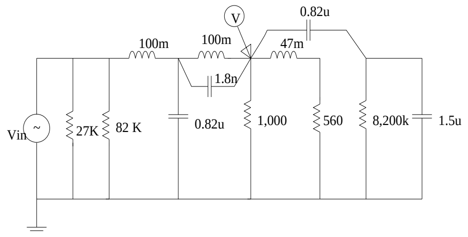

Figure 13: Best evolved low-pass filter using the ILG approach

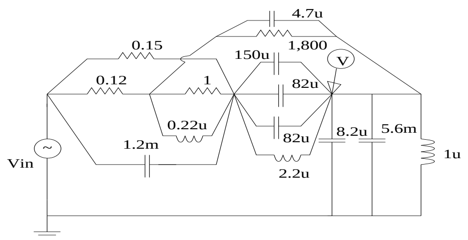

Figure 14: Best evolved band-pass filter using the ILG approach

reject the ones below 1,000 Hz and above 4,000 Hz. The notch filter has been specified to accept frequencies below 1,000 Hz and above 5,000 Hz, rejecting the ones between 2,000 and 4,000 Hz. Figures 13, 14 and 15 show the schematics of the best low-pass (fitness of 981 over 1,000), band-pass (fitness of 928 over 1,000) and notch (fitness of 878 over 1,000) filters; Figure 16 shows their respective frequency responses.

It has been verified that the multiplication of the $change - mask$ operator rate by a decreasing exponential factor allowed the ILG strategy to arrive at smaller circuits. So, in this class of experiments, the general expression for this operator rate has been set to $C.e^{-kx}$, where x is the current generation number, C was set to 0.2 and k was set to 0.005. In the case of the UDIP strategy this rate was set constant to the value of 0.03 (3% of probability per mask bit).

Let us compare our results for the evolution of the low-pass filter (also called "Brick Wall" filter) with the genetic programming methodology [21]. Using Genetic Programming, a fully compliant

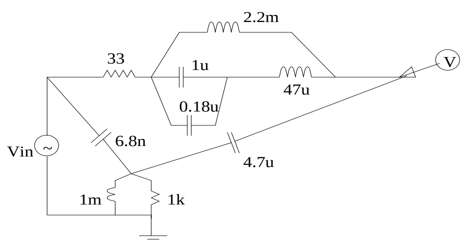

Figure 15: Best evolved notch filter using the ILG approach

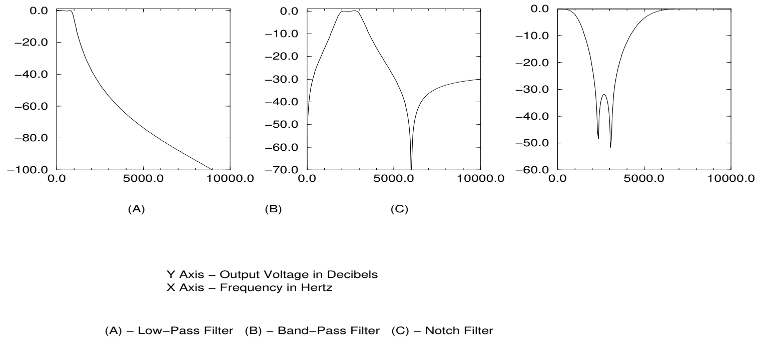

Figure 16: Frequency responses of the passive filters

circuit has been evolved (Expression 7 equal to zero), while our VLR methodology and the ILG strategy have arrived very nearly to the fully compliant circuit (fitness of 981 over 1,000). Nevertheless, our VLR methodology arrived at a slightly more parsimonious circuit (12 components against 16 in the GP methodology [21]). Additionally, the total number of individuals processed within our framework has been much smaller than the one in GP: 100,000 (40 individuals, 500 generations and 5 executions) in our methodology against 10,240,000 (320,000 individuals and 32 generations) in GP.

The EA used in these case studies have not used the memory paradigm. In order to use this paradigm, we had to change our representation, so that each gene encodes for a circuit mesh, whose performance may be assessed. So, the topologies are now constrained to circuits of a variable number of parallel meshes [37]. We made experiments for the particular case of the band-pass filter defined above, using the ILG methodology. We stored in the memory genes encoding for meshes behaving as either low-pass filters or high-pass filters. This kind of circuits can be useful building blocks for the evolution of the band-pass filter. Figure 17 shows the best band-pass filter evolved using the memory paradigm; in this circuit, the authors identified one explicit building block, found both in the memory and in the best circuit. Figure 18 shows the frequency response of the whole circuit and of the explicit building block (a high-pass filter with cut-off frequency below 100 Hz). Table 4 compares the following results: with and without the memory paradigm and constraining the circuit topologies to meshes (over 10 executions); and the previous results allowing any kind of topologies. The results suggest that the memory paradigm is improving performance; though, opposing to their digital counterparts, in which there is no epistasis effect among genes, in this case the individual meshes will affect each other to a certain extent, due to electric coupling. Therefore, more research needs to be done to arrive at the least epistatic representation.

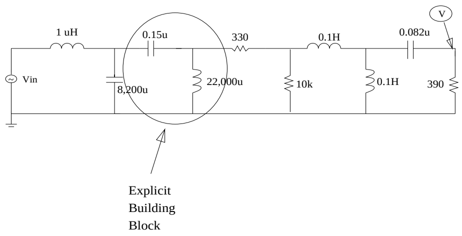

Figure 17: Evolved band-pass filter using the memory paradigm (fitness of 862 over 1,000)

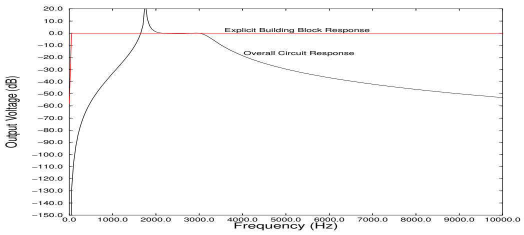

Figure 18: Frequency responses for the circuits shown in Fig. 17

Table 4: Results on the use of the memory paradigm on analog evolution

Methodology	$< r >$	r_{best}
Using the Memory Paradigm with Mesh Representation	114.7	107.9
Without the Memory Paradigm with Mesh Representation	93.9	103.8
Without the Memory Paradigm without Mesh Representation	98.1	66.3

Table 5: Results on Analog Amplifiers Evolution - UDIP and OLG strategies using the Pareto control technique

Methodology	Statistics	Performance
ILG	$< r >$	0.3
ILG	r_{best}	1.04
UDIP	$< r >$	0.09
UDIP	r_{best}	0.31
OLG	$< r >$	0
OLG	r_{best}	0

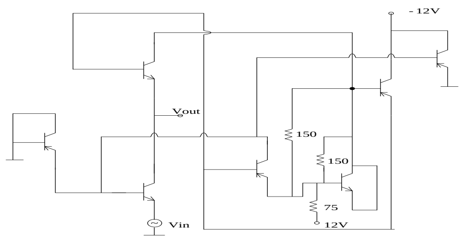

Figure 19: Schematic of the best amplifier obtained by the ILG methodology

4.2.2 Transistor Based Amplifiers

The design of analog amplifiers is usually a complex task and can be regarded as being more a form of art than of science, requiring experience and intuition from the designer [23]. Koza et. al. [20] have published pioneering applications, applying Genetic Programming in the design of amplifiers. We have used our VLR methodologies to tackle this problem. The same representation of the passive filters application has been used, except that now each gene may encode a npn transistor, a pnp transistor or a resistor. The gene also informs the components' connecting points and their magnitude, in case it is a resistor. The fitness evaluation function is made through the evaluation of the DC response of the circuit, as suggested in [2]. The SMASH circuit simulator has also been used in this case study. After applying a small input signal, of the order of mV, we measure the induced variation in the output voltage. The higher the output variation, the better the amplifier.

Table 5 compares the strategies' performances; the minimum specified fitness threshold T has been set to 20 dB gain amplifiers. Both the first and second methodologies comply with this specification. Again, it can be verified that the ILG strategy outperformed the others.

Figure 19 shows the schematic of a 55 dB gain amplifier obtained by our methodology using the ILG strategy and Figure 20 shows its DC response to a 20 mV input signal. This figure shows both the response obtained in simulation and the behaviour of the circuit when **implemented in reality**, demonstrating that the results can be reproduced in practice.

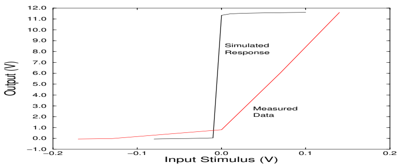

Figure 20: DC response of the circuit in Figure 19

Using the genetic programming methodology [2], a 60 dB gain circuit with 22 transistors has been evolved, while we arrived at a 55 dB gain circuit (measured in simulation), with 7 transistors. In [2], linearity has also been taken into account in the fitness function. We have processed a total of 80,000 individuals (40 individuals, 200 generations and 10 executions, each one taking around 4 hours in a Sparc Ultra 2) against 69,760,000 in GP (109 generations and 640,000 individuals). We also ensured the testability of our circuits by mounting them in protoboards.

5 Discussion: Biological Genome Evolution

The concepts presented in this section are a brief overview on some aspects of biological evolution, concerning to variable length representation; for further details, report to [28]; we discuss the analogies of these biological concepts with our VLR methodologies.

Regarding to variable length representation in biology, geneticists still disagree about the proportion of non-coding DNA in the human genome; estimates, made by DNA re-association experiments, vary from 10% to 50%. The human genome contains around 100,000 genes, meaning that a human being uses about 100,000 different kinds of proteins in his life. As a protein is made up of an average of 300 amino-acids, which leads to about 1,000 nucleotides, one concludes that there are around 10^8 coding bases in the human genome. Notwithstanding, as mentioned before, a human cell contains about 3 x 10^9 nucleotide units, much more than the amount needed to code for proteins.

In spite of the imprecision, it is certain that most DNA does not code for genes, being called non-coding DNA. Table 6 illustrates one experiment, displaying data for both vertebrate and invertebrate species [28]. For each species example, the table shows the amount of DNA, expressed as haploid DNA content in picogram and the percentage of coding DNA. For each species we picked out a particular example to show the statistics, but there is a large amount of variation within a single species.

From this table, one can verify the large differences among species. For instance, the bacterial DNA, Tetrahymena pyriformis, is very efficient, using most of its DNA. Also, from looking at this table, we see that the amount of DNA in a salamander (Necturus Masculosus) is 20 times than that in a human being. As remarked in [28], it is implausible that the number of radically different

Table 6: Amount of DNA and Estimated Percentage of Coding DNA (extracted from [28])

Species	Amount of DNA (pg)	Estimated Coding DNA (%)
Protozoa (Tetrahymena pyriformis)	0.2	90
Mollusca (Loligo loligo)	2.8	75
Arthropoda (Limulus Poliphemus)	2.8	70
Pisces (Leuascus cephalus)	5.5	44
Amphibia (Necturus Masculosus)	83.0	12
Reptilia (Natrix natrix)	2.5	47
Aves (Gallus domesticus)	1.2	80
Mammalia (Homo sapiens)	3.5	64

genes needed in a salamander is 20 times than that in a man. The fact that the percentage of coding DNA in a human being is much higher than in a salamander provides an explanation for this contradiction.

It has also been verified that the non-coding DNA is highly repetitive. There are two kinds of repetitive sequences: tandem and scattered repeats. In the former the repeat sequences appear next to each other and in the latter they are scattered through the genome.

Two main evolutionary hypotheses have been suggested to explain the presence of this repetitive, non-coding DNA. One hypothesis says that such DNA is functional, even though it does not actually encode for genes. It may be needed for some regulatory or structural reason not discovered yet. Alternatively, the repetitive DNA may be selfish DNA - either neutral 'junk' DNA, parasitic DNA or introns. The selfish DNA hypothesis is the one that is most accepted by the geneticist community. The selfish DNA may also be active or passive. They are considered active when they proliferate through the DNA until checked by natural selection because they interfered with vital DNA functions or because they became so abundant that their replication imposed a significant cost to the organism.

There are three mechanisms that may have been creating non-coding DNA along the evolutionary process: unequal crossover, slippage and rolling circle. Unequal crossover produces strands with a greater or lesser number of repeats and successive rounds of unequal crossover can build up or break down long sequences of repeat. Slippage occurs when nucleotides are missed or copied twice in the process of DNA strand replication. Finally, rolling circle replication can occur whenever the sequences form a circle and multiply itself; this has been observed in virus and bacterias, but not in eukaryotes.

Besides the fact that our model is a great simplification of natural mechanisms, there is another major difference between the two: macroevolutionary changes in nature are carried out through adaptation and gradual changes, and not through genome expansion, as defined in our model. Nonetheless, our model borrows some principles from nature, that are not included in standard EAs: increasing growth in phenotype complexity, the accessibility of the effect of each gene in the phenotype and a higher degree of preservation of useful genetic material.

6 Conclusions

We have demonstrated in this work the high level of correlation between two areas of study: evolutionary electronics and variable length representation evolutionary algorithms. The theoretical methodologies from the latter can be used in the former as a large source of case studies; the former, on the other hand, may produce many results of practical importance for CAD development by borrowing concepts from the latter.

We have made an extent study of cases, including analog and digital electronics, covering most of the evolutionary electronics applications being investigated in the present moment. Along our experiments, we have made comparisons on the performance of different strategies for sweeping the VLR genome space, investigated implicit and explicit mechanisms for balancing accuracy and parsimony, and proposed an explicit way of using non-coding segments.

It has been verified that the increasing length genotypes strategy for sweeping the genome space produced the best results for the kind of engineering applications developed here, mainly due to its inherent implicit mechanism for controlling the solutions' size.

A technique based on the Pareto concept for multi-objective optimisation has been applied as an explicit way for balancing parsimony and accuracy, but it did not produce any visible improvement in the results. The main effect of using this mechanism is to indirectly increase the selection pressure towards parsimonious and fit solutions; as in evolutionary electronics it is very likely that long genotypes produced initially will have very low fitness (usually not even being simulatable), explicit balance control mechanisms are usually redundant.

The use of the memory paradigm allowed the exploration of non-coding segments and has produced significant benefits in the evolution of large combinational functions. Even though this kind of non-coding material is usually seen as a problem in genetic programming, they suit very well to the application of the technique developed by the authors. The main challenge to get the benefits of the memory paradigm in a broader domain of applications is in finding representations with low level of epistasis among genes. The authors are now investigating the application of this concept in the problem of VLSI placement [8].

The authors see as future works the following topics: extension of the applications on combinational evolution to handle circuits of larger sizes; further investigation on the use of the memory paradigm on analog circuits evolution; producing professional CAD tools for analog circuit synthesis using evolutionary algorithms.

7 Appendix - Evolutionary Electronics Genome Space

This section estimates the average fitness of the genome space as long as the genotypes grow in size, for the special case of analog evolutionary electronics. If a particular genotype has its upper size limited to n and each gene can take a total of k values, depending on the contents of its loci, the following equation gives the number of genotypes G that code for a particular target circuit with p components, where each gene codes for one component:

$$\mathrm{G} = \frac{n!}{(n-p)!} k^{l} (k-p)^{(n-p-l)} U_{n-p-l}(l) \tag{8}$$

,over a genome space of size k^n x 2^n (possible combination of main and auxiliar mask strings). The above equation is based on the assumption that a genotype will encode the particular target circuit if and only if p of its n genes assume a particular set of values, p-set, regardless of the order in which it appears. The other $n-p$ genes are not contributing effectively to the phenotype and they can be either activated or deactivated, so that $(n-p) \geq l$ (refer to equation 2). The fraction at the left of equation 8 accounts for the number of different places the p-set can take within the genotype, i.e., the arrange of p over n positions (we made the simplification that the p genes will have distinct values); the second term accounts for the different values the genes that are not contributing to the phenotype and are deactivated can take; the third term accounts for the different values the genes that are not contributing to the phenotype and are activated can take; the last term is a step function stating that, if the number of deactivated genes, l, is bigger than $n-p$, then no genotype may encode for the structure. We assume that the $n-p-l$ genes that are activated but are not contributing to the phenotype are not allowed to take any of the p values used to encode for the circuit; if they could assume one of these values, then they would be contributing to the phenotype, because each gene determines the connecting points of the component in the circuit. The effect of these non-contributing genes would be of components that are not connected to the main circuit, in case they are active. Equation 9 gives then the density of a particular structure encoded by p genes over the genome space of maximum dimensionality n:

$$\mathrm{D} = \frac{n!}{(n-p)!} k^{l} (k-p)^{(n-p-l)} U_{n-p-l}(l) \frac{1}{k^n 2^n} \tag{9}$$

In order to get a feeling of how this calculations will fit into a practical evolutionary electronics case, we can consider the issue of evolving an operational amplifier within a connecting points matrix of size 20. This is shown in Figure 21. The 20 points matrix has 5 points which serve as connections to the input signal and power supplies, and one particular point which is probed as output. Each gene will have four loci to encode for one transistor; one locus selecting between a p-transistor and a n-transistor, and the other three, selecting among the 20 points of the matrix, the three transistor terminals. Thus, the total number of values each gene can take, k, is 2 x 20^3 = 16,000. The operational amplifier is composed of three stages. If we regard the performance as

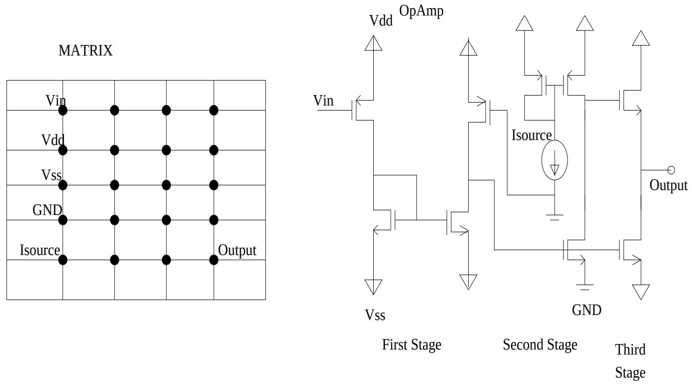

Figure 21: Evolution of an OpAmp in a 20 points matrix

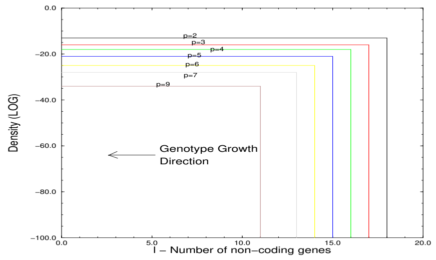

Figure 22: Density of non-zero fitness structures

being measured by the amplification capacity, given by the first two stages, and by the buffering capacity, given by the third stage, than not only the overall three stage structure would assign a good fitness, but also each of the three stages and combinations of two stages, resulting in seven nonzero fitness structures, with p=2, p=3, p=4, p=5, p=6, p=7 and p=9.

Supposing that the seven structures score the same fitness, we will have the behaviour shown in the graph of Figure 22, in which we plot the density curves for all structures according to equation 9 (for n=20). It can be seen that the density of each emerging structure rises sharply when the genome size reaches its necessary number of genes, and then stays constant. The main conclusion drawn from this model is that, **once the necessary amount of genetic material that encodes for a particular structure is achieved, there is no need to sample more complex regions of the genome space, since the density function remains constant.** This model supports the graph of Figure 2, if we aggregate over a large amount of emerging structures.

If we take into account the fact that, as we move toward more complex regions of the genome space, it is more likely to find a larger number of structures behaving as amplifiers, we can expect that the constant density value achieved for bigger values of p will be higher than the ones forecasted in our model.

8 Acknowledgements

The authors wish to thank CAPES, Brazilian Federal Agency, for the financial support, and Dr. Adrian Thompson, Dr. Inman Harvey and Dr. David Nicholson for the collaboration.

References

[1] Banzhaf, W., Francone, F., Nordin, P. , "On some emergent properties of variable size evolutionary algorithms", in http://www.ai.mit.edu/people/unamay/icga-submissions.html, ICGA'97 Workshop on Evolutionary Computation with Variable Size Representation. July, 20, 1997.

[2] Bennett III, F. H., Koza, J. R., Andre, D., Keane, M. A., " Evolution of a 60 Decibel Op Amp Using Genetic Programming", Proceedings of the First International Conference on Evolvable Systems (ICES96), LNCS 1259, pp. 455-469, Tsukuba, Japan, October, 1996.

[3] Bishop, O., "Practical Electronic Filters", Babani Electronics Books, 1991.

[4] Blickle, T., " Theory of Evolutionary Algorithms and Application to System Synthesis", PhD dissertation, Swiss Federal Institute of Technology, Zurich, 1996.

[5] Brayton, R. K., Hatchel, G. D., McMullen, C., T., Sangiovanni-Vincentelli, A., "ESPRESSO-II: a new logic minimiser for programmable logic arrays", IEEE Custom Integrated Circuits Conference, Rochester, New York, pp. 370-376

[6] Dreschler, R., Becker, B., Gockel, N., "A Genetic Algorithm for minimization of Fixed Polarity Reed-Muller expressions", in the International Conference on Artificial Neural Networks and Genetic Algorithms, pages 392-395, 1995.

[7] Esbensen, H., "A Macro-cell global router based on two genetic algorithms", in European Design Automation Conference, pages 428-433, 1994.

[8] Esbensen, H., Mazumder, P., "SAGA: A Unification of the genetic algorithm with simulated annealing and its application to macro-cell placement", in VLSI esign Conference, pp. 211-214, 1994

[9] Fonseca, C. M, Fleming P. J. , "An Overview of Evolutionary Algorithms in Multiobjective Optimization", Evolutionary Computation, Vol. 3, No.1, pp.1-16, 1995.

[10] Gockel, N., Dreschler, R., Becker, B., "Learning Heuristics by Evolutionary Algorithms with Variable Size Representation", in http://www.ai.mit.edu/people/unamay/icga-submissions.html, ICGA'97 Workshop on Evolutionary Computation with Variable Size Representation. July, 20, 1997.

[11] Goldberg, D., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[12] Goldberg, D. E., Deb, K., Korb, D., "An Investigation of Messy Genetic Algorithms", Technical Report TCGA 90005, TCGA, the University of Alabama, 1990.

[13] Grimbleby, J. B. , "Automatic Analogue Network Synthesis Using Genetic Algorithms", Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems (GALESIAS - 95), pp.53-58, UK, 1995.

[14] Harvey, I., Thompson, A., "Through the Labyrinth Evolution Finds a Way: A Silicon Ridge", Proceedings of the First International Conference on Evolvable Systems (ICES96), LNCS 1259, pp. 406-422, Tsukuba, Japan, October, 1996.

[15] Harvey, I., "The Artificial Evolution of Adaptive Behaviour", PhD Thesis, University of Sussex, School of Cognitive and Computing Sciences, September, 1993.

[16] Horrocks, D. H, Spittle, M. C., "Component Value Selection For Activate Filters Using Genetic Algorithms", 1st On-line Workshop on Soft Computing (WSC1), Special Session on Evolutionary Electronics, August 19-30, 1996.

[17] Iba, H., Iwata, M., Higuchi, T., "Machine Learning Approach to Gate-Level Evolvable Hardware", Proceedings of the First Conference on Evolvable Systems: From Biology to Hardware, LNCS 1259, pp. 327-343, 1997.

[18] Kommu, V., Pomenraz, I., "GAFAP: Genetic Algorithm for FPGA technology mapping", in European Design Automation Conference, pp. 300-305, 1993

[19] Koza, J. R., "Genetic Programming On the Programming of Computers by Means of Natural Selection", MIT Press, 1992.

[20] Koza J. R., Bennett III F. H., Andre, D. , Keane, M. A., "Four Problems for which a Computer Problem Evolved By Genetic Programming is Competitive with Human Performance", Proc. of 1996 IEEE International Conference on Evolutionary Computation, IEEE Press., Pages 1-10.

[21] Koza J. R., Bennett III F. H., Andre, D. , Keane, M. A, "Toward Evolution of Electronic Animals Using Genetic Programming", Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems, Cambridge, MA: The MIT Press.

[22] Koza, J. R., "Genetic Programming II Automatic Discovery of Reusable Programs", MIT Press, 1994.

[23] Koza, John R., Bennett III, Forrest H, Keane, Martin A., and Andre, David. 1997b. Evolution of a time-optimal fly-to controller circuit using genetic programming. In Koza, John R., Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max, Iba, Hitoshi, and Riolo, Rick L. (editors). 1997. Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13shyp;16, 1997, Stanford University, San Francisco, CA: Morgan Kaufmann. 207 shyp; 212.

[24] Kruiskamp, W. and Leenaerts, D., "DARWIN: CMOS opamp synthesis by means of genetic algorithm", Proceedings of the 32nd Design Automation Conference, New York, NY: Association for Computer Machinery, pp. 698-703, 1995.

[25] Langdon, W., "Fitness causes bloat in variable size representations", in http://www.ai.mit.edu/people/unamay/icga-submissions.html, ICGA'97 Workshop on Evolutionary Computation with Variable Size Representation. July, 20, 1997.

[26] Miller, J. F., Thomson, P. , "Combinational and Sequential Logic Optimization Using Genetic Algorithms", Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems (GALESIAS - 95), pp.34-38, UK, 1995.

[27] Nordin, P., Francone, F., Banzhaf, W., "Explicitly defined introns and destructive crossover in genetic programming". In Justinian P. Rosca, editor, Proceedings of the Workshop on Genetic Programming (at Twelfth Int. Conf. on Machine Learning), pages 6-22, Technical Report 95.2, University of Rochester, 1995

[28] Ridley, M., "Evolution", Second Edition, Blackwell Science, USA, 1996.

[29] Rogenmoser, R. , Kaeslin, H. , Blickle, T. , "Stochastic Methods for Transistor Size Optimization of CMOS VLSI Circuits", in Parallel Problem Solving from Nature IV, pp. 849-858, Springer-Verlag, LNCS 1141, 1996.

[30] Sakanashi, H., Higuchi, T., Iba, H., Kahazu, Y., "Evolution of Binary Decision Diagrams for Digital Circuit Design Using Genetic Programming", Proceedings of the First Conference on Evolvable Systems: From Biology to Hardware, LNCS 1259, pp. 470-481, 1997.

[31] "SMASH User and Reference Manual", Dolphin Integration, France, 1993.

[32] Smith, S. F., "A Learning System Based on Genetic Adaptive Algorithms", PhD Thesis, Department of Computer Science, University of Pittsburgh, USA, 1980.

[33] Thompson, A., "An Evolved Circuit, Intrinsic in Silicon, entwined with physics", Proceedings of the First International Conference on Evolvable Systems (ICES96), Tsukuba, LNCS 1259, pp. 390-405, Japan, October, 1996.

[34] Thompson, A., "Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution ", PhD Thesis, University of Sussex, School of Cognitive and Computing Sciences, September, 1996.

[35] Thompson, A., Harvey, I., Husbands, P., "Unconstrained Evolution and Hard Consequences", in "Towards Evolvable Hardware: An International Workshop," Lausanne, Switzerland, October 2-3, 1995. Published as a chapter in "Towards Evolvable Hardware: The evolutionary engineering approach," pages 136-165, edited by E. Sanchez and M. Tomassini, Springer-Verlag LNCS 1062, 1996.

[36] Zebulum, R. S., Pacheco, M. A., Vellasco, M., "Evolvable Systems in Hardware Design: Taxonomy, Survey and Applications", Proceedings of the First Conference on Evolvable Systems: From Biology to Hardware, LNCS 1259, pp. 344-358, 1997.

[37] Zebulum, R. S., Pacheco, M. A., Vellasco, M., "Comparison of Different Evolutionary Methodologies Applied to Electronic Filter Design", to be published in the Proceedings of the IEEE International Conference on Evolutionary Computation, to be held in Alaska, May 2-4, 1998.

[38] Zebulum, R. S., Pacheco, M. A., Vellasco, M., "Increasing Length Genotypes in Evolutionary Electronics", in http://www.ai.mit.edu/people/unamay/icga-submissions.html, ICGA'97 Workshop on Evolutionary Computation with Variable Size Representation. July, 20, 1997.

[39] Zhang, B., Muhlenbein, H., "Balancing Accuracy and Parsimony in Genetic Programming", Evolutionary Computation, 3(1), pp. 17-38, 1995.

[40] Zhang, B., "A taxonomy of control schemes for genetic code growth", in http://www.ai.mit.edu/people/unamay/icga-submissions.html, ICGA'97 Workshop on Evolutionary Computation with Variable Size Representation. July, 20, 1997.