Genetic Programming with Genetic Regulatory Networks

Genetic Programming

Rui L. Lopes
Centro de Informática e Sistemas da
Universidade de Coimbra
Polo II - Pinhal de Marrocos
3030-290 Coimbra, Portugal
rmlopes@dei.uc.pt

Ernesto Costa
Centro de Informática e Sistemas da
Universidade de Coimbra
Polo II - Pinhal de Marrocos
3030-290 Coimbra, Portugal
ernesto@dei.uc.pt

ABSTRACT

Evolutionary Algorithms (EA) approach differently from nature the genotype - phenotype relationship, and this view is a recurrent issue among researchers. Recently, some researchers have started exploring computationally the new comprehension of the multitude of regulatory mechanisms that are fundamental in both processes of inheritance and of development in natural systems, by trying to include those mechanisms in the EAs.

One of the first successful proposals was the Artificial Regulatory Network (ARN) model. Soon after some variants of the ARN, including different improvements over the base model, were tested. In this paper, we combine two of those alternatives, demonstrating experimentally how the resulting model can deal with complex problems, including those that have multiple outputs. The efficacy and efficiency of this variant are tested experimentally using two benchmark problems that show how we can evolve a controller or an artificial artist.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic Methods

General Terms

Algorithms

Keywords

artificial regulatory network, genetic programming, inverted pendulum, artificial art

1. INTRODUCTION

Nature-inspired algorithms are used extensively today to solve a multitude of learning, design, and optimisation problems, giving rise to a new research area called Evolutionary Computation (EC) [1]. Over time many variants of a basic algorithm especially tuned for some problems and/or situations were proposed (e.g., algorithms for dealing with noisy, uncertain or dynamic environments, for evolving rather than designing the algorithm's parameters or some of its components, algorithms with local search operators or for multi-objective optimisation). Typically, the objects manipulated by the evolutionary algorithms are represented at two different levels. At a low level (the genotype) the representations are manipulated by the variation operators; at a high level (the phenotype) the objects are evaluated to determine their fitness and are selected accordingly. Because of that, we need a mapping between these two levels. The issue of the relationship between the genotype and the phenotype is as old as the area itself, with many experts claiming that the standard approach is too simplistic, and advocating that it is worth trying to close the gap between EC and Biology, introducing more complex relationships between the genotype and the phenotype, and evaluate if they are responsible for an improvement in both robustness and evolvability [2, 3]. For example, typically in an EA, the two phases of transcription and translation (that exist in nature) are merged into just one and the regulatory processes are missing. At a larger scale, we could add the lack of epigenetic phenomena that contribute to the evolution and all the mechanisms involved in the construction of an organism.

During the last years we saw the appearance of novel representations and the corresponding genotype to phenotype mapping: grammatical evolution [4], self-modifying cartesian genetic programming [5], gene expression programming [6], or enzyme programming [7]. Along a different path, W. Banzhaf et al. [3], suggested that one should enrich the artificial model of evolution with the inclusion of feedback regulatory mechanisms. Earlier, in [8] the author had proposed an artificial gene regulatory network(ARN) model and showed how it could be used computationally in different settings [9]. Later, the ARN model was extended by [10] with the inclusion of extra input proteins and dividing the gene's products in transcription factors and non-regulatory proteins. The latter are used as outputs. More recently [11] presented another variant of the ARN, enlarging its applicability, by transforming the regulatory gene network into a computable graph, with or without feedback edges, similarly to what is done in GP. In this work we merge these two approaches and explore further its capabilities.The combined model is tested with two benchmark problems, one involving the evolution of a controller and the other an artificial

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

artist. The latter is developed as a proof-of-concept for the model's multiple output capabilities.

The paper is organised as follows. Section 2 is a brief overview of the concepts behind regulation at the transcriptional and translational levels, the types of models that have been proposed, and the contribution of evolutionary computation to the problem of inferring GRNs from data. Section 3 describes the original ARN model as originally proposed by the author. Then, Section 4 describes an extension of that model, elucidating in particular how a program can be extracted from a network. In Section 5 we briefly refer to the problems used and we present the experimental setup in Section 6. The results are presented and analysed in Section 7. Finally, in Section 8 we draw some conclusions and present ideas for future work.

2. GENE REGULATORY NETWORKS

For a long time it was believed that the DNA was transcribed into RNA, which in turn was translated into proteins in a one-way process. Today, we know that the process of gene expression into proteins is more complex, and relies on a network of interactions (known as regulatory network) between genes and many other molecules including proteins, the very products of gene expression (see Fig. 1).

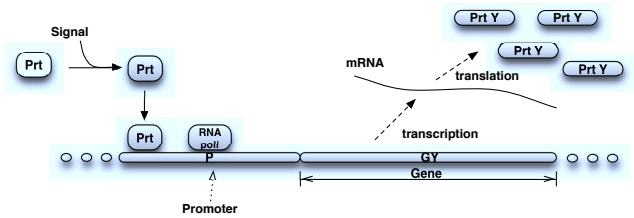

Figure 1: Gene Regulation: the transcription of a gene is driven by a protein called RNA polymerase, which binds to a special regulatory region of the DNA called the promoter. Other proteins (transcription factors) control the accessibility of the promoter to RNA polymerase. These proteins are activated by external signals and bind in the vicinity of the promoter. They can enhance the production of a protein by a gene, as in the figure, or inhibit its production.

A correct understanding of the behaviour of gene regulatory networks (GRN) is of paramount importance in many areas of application, like personalised medicine, epidemic prevention, energy production, bio-remediation or synthetic biology [12].

Various approaches for formally modelling gene regulatory networks appeared in the last decades. The proposed models can be classified according to the following aspects: variables such as product concentrations are discrete, continuous or mixed; time is discrete and the update of the variables is either synchronous or asynchronous (there are, however, cases where time is continuous); space is discrete, continuous or absent. Examples of models include directed graphs, bayesian networks, ordinary and partial differential equations, random boolean networks, neural networks, and rule-based formalisms [13, 12, 14, 15]. Since these regulatory networks are highly non-linear and have several thousand variables obtaining a model is a complex task. Computational approaches to this problem try to reconstruct the GRN from experimental data, for instance, gene expression data obtained from microarrays. Within this perspective devising a GRN is viewed as a reverse engineering problem, where one builds a model from data. Some attempts were made to use evolutionary algorithms to solve this problem for some of the formal models. In particular, there have been proposed solutions based on genetic algorithms [16], genetic programming [17], evolution strategies [18], and differential evolution [19].

Researchers have been interested in GRNs from different points of view, besides the biological one. For example, the different artificial models have been studied from a system's perspective, trying to understand its topological and dynamics properties [20, 21, 22, 9]. In a different path, some computational explorations have been proposed that reflect our comprehension about regulatory mechanisms and how they mediate between evolution and development, i.e., that try to incorporate those mechanisms into artificial evolutionary systems. For example, in [23, 24, 25, 26, 27] GRNs are explored in the context of artificial 3D morphogenesis. In [28, 29] regulation is explored to evolve real-time controllers for a robot, and to the task of automatically designing robots in a physical-based virtual environment. In [20] the authors proposed a model of an artificial genome and studied the implications for artificial ontogeny, while [30] also proposed a model of gene expression and regulation in an artificial cellular organism and speculates about its application to evolutionary computation. There is a third perspective for studying artificial GRNs: using them as a computational device. To the best of our knowledge, the unique proposal of an artificial regulatory network used for problem solving, was the one proposed in [8] (ARN). Due to its importance for our work it will be detailed in the next section.

3. THE ARN MODEL

The Artificial Regulatory Network (ARN) [8] is an attempt to incorporate regulatory mechanisms between the genotype and the phenotype. There are no other products, i.e., DNA, and processes in between these two levels. The genome has fixed length and is constructed by simple duplication with mutation events. Regulation between genes is a mediated process, achieved by means of a binding process between proteins (i.e., transcription factors) and special zones in the genome that appear upstream of the promoter of a gene. The remaining of this section will describe this with more detail.

Genome

The model presented in this paper uses a binary genome and implements a simple algorithm to transcribe and then translate it into proteins. The genome can be generated randomly or by a process of duplication with mutation, also called *DM*, that is considered the driving force for creating new genes in biological genomes and has an important role in the growth of gene regulatory networks [31]. In the latter case we start with a random 32-bit binary sequence, that is followed by several DM episodes. As we will see later the number of duplications is an important parameter. So, if we have 10 duplication events then the final length of the genome is $2^5 \times 2^{10} = 32768$. The mutation rate is typically of 1%. The genome is divided in several regions, namely a regulatory site, the promoter, and the gene itself. The first

32 bits of the regulation zone are the enhancer site, while the following 32 bits are the inhibitory site. The promoter is located downstream and has the form $XYZ01010101$.The first 24 bits (the sequence represented by XYZ) can be either 0 or 1, while the last 8 bits are fixed. This way, the probability for a promoter to occur is $2^{-8} = 0,39\%$. The idea is to model what happens in nature, where we have a small consensus sequence where the RNA polymerase binds (in our case 01010101), inside a larger promoter. A gene is composed of five 32-bit long sequences, i.e., a 160-bit string. The choice of the method to obtain the genome, including the values for the parameters, is guided by what happens in the natural world.

Gene expression

The genotype - phenotype mapping is defined by expressing each 160-bit long gene, resulting in a 32-bit protein. A gene expresses a protein by a majority rule: if we consider a gene, for example G^m, divided into 5 parts of size 32 each, G^m_1 to G^m_5, at position i, say, the protein's bit will have a value corresponding to the most frequent value in each of these 5 parts, at the same position, i.e.,

$$P^m_i = majority(G^m_{ki}, \quad \forall k = 1, \ldots, 5), \quad \forall i = 1, \ldots, 32$$

Figure 2 show a simple illustrative example.

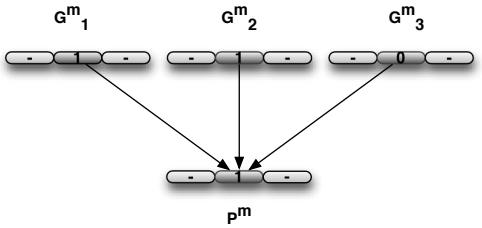

Figure 2: The majority rule: a simple example for a gene with three regions of size three and the corresponding protein. Here just the process for the second position.

Figure 3 gives an idea of the representation. P is the promoter region, that indicates the beginning of the gene; G1 to G5 are the five parts of a gene; E is the gene activation binding site (enhancer) for the protein, and H is the repression binding site for the protein (inhibitor).

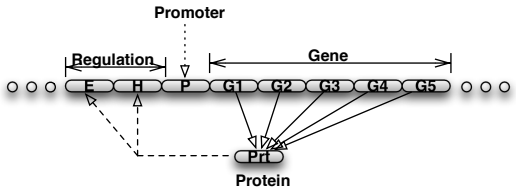

Figure 3: A genome element in the ARN model.

Regulation

Genes interact mediated by proteins, which bind to the regulatory region of each gene. If, say, gene **A** expresses protein p_A and that protein contributes to the activation of gene **B**, we say that gene **A** regulates **B** (see Fig. 4).

Notice that in order for a link to exist between any two genes, the concentration of the corresponding protein must

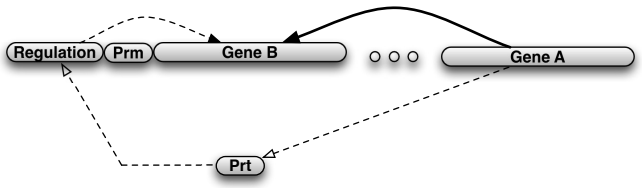

Figure 4: Gene - Protein - Gene interaction

attain a certain level, and that depends on the strength of the binding. The strength of the binding is computed by calculating the degree of complementarity between the protein and each of the regulatory regions, according to formula 1:

$$x_i = \frac{1}{N} \sum_{j=1}^{N} c_j e^{\beta(\mu_{ji} - \mu_{max})} \tag{1}$$

where x_i represents the binding strength of the enhancer (e_i) or the inhibitory (h_i) region, N is the number of proteins, c_j the concentration of protein j, μ_{ji} is the number of bits that are different in the protein (j) and in the regulation site (r_i), that is,

$$\mu_{ji} = \sum_{k=1}^{32} j_k \oplus r_{i_k} \tag{2}$$

μ_{max} is the maximum match achievable, and β is a scaling factor. The production of a protein over time depends on its concentration, which in turn is a function of the way each protein binds to that gene's regulatory regions. It is defined by the differential equation

$$\frac{dc_i}{dt} = \delta(e_i - h_i)c_i \tag{3}$$

where e_i and h_i are defined by equation 1, and δ is a scaling factor.

Computational Device

Using this process we can build for each genome the corresponding artificial gene regulatory network. From a problem-solving perspective we want to transform an ARN into a computational problem-solver.To that end we need to clarify what we put into the system, what we extract from the system, and how we define the semantics, that is, the meaning of the computation in which the network is engaged. Finally, and as a consequence of the points just identified, it is also fundamental to determine if we are interested in the input/output relationship or if what we want is just the output. A solution for the latter situation was proposed in [32] in the context of optimization problems. The idea is to define (randomly) two new contiguous 32-bit sequences in the genome. The first one being a new inhibitory site (h_i), and the second one a new activation site (e_i). All generated proteins can bind to these sites. The levels of activation and inhibition can be computed as before (Equation 1), but there is no gene (thus no protein) attached.

The state of this site is just the sum of all bindings (see Eq. 4) and is defined as the output. This additional binding is thus a method to extract a meaning from the variation of

the proteins' concentrations over time.

$$s(t) = \sum_i (e_i - h_i) \quad (4)$$

To use the model as a representation formalism for genetic programming one needs to define what are the inputs and what are the outputs. For that purpose the ARNs model was extended in two directions [10]. First, some extra proteins, not produced by genes but contributing to regulation, were introduced and act as inputs. Second, the genes were divided into two sets, one producing proteins that are used in regulation (i.e., transcriptional factors), and a second one with proteins without regulatory function which are used as outputs. These two types of genes are distinguished by having different promoters (see Fig. 5).

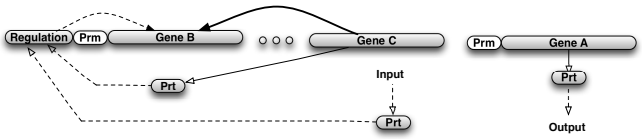

Figure 5: The modified ARN proposed in [10].

This model has been successfully used to solve different problems, namely, the cart-pole balancing [10] and to predict time-series [33]. A different approach presented in [11], called ReNCoDe, employs the original ARN architecture as the genotypical representation, without using the proteins' concentrations, and proposed an algorithm to extract a program from the regulatory network. Moreover, new biologically inspired genetic operators were introduced, which improved the performance of the algorithm. In particular the *transposon* operator, which copies a section of the genome and inserts it into a new random location. In the current paper we propose to merge these approaches. We keep the fundamental aspects of ReNCoDe, i.e., the transformation of the GRN in a program that ignores the protein's concentrations, and the new genetic operators, but introduce extra input proteins and output proteins. This reformulation will be described in the following sections.

4. GENETIC PROGRAMMING WITH ARNS

The merge of ReNCoDe [11] with the extended ARN model presented in [10] involves three aspects: the definition of the inputs and the outputs, the algorithm for extracting the program from the network, and the mapping of proteins to functions.

4.1 Input and Output

Extra proteins are introduced to represent the inputs and the proteins coded in the genome are distinguished between transcription factors and products, based on using two promoters $XYZ00000000$ and $XYZ11111111$, respectively. The extra proteins have distinct 32-bit signatures. The products are regulated by all proteins but do not regulate, while the inputs participate in the regulatory process but are not regulated. This means that, in the resulting network, there are not connections towards the inputs, nor from the output(s) towards other proteins (see Fig. 5).

4.2 Building the graph

First the ARN of the individual is built, composed of multiple links (inhibition and excitation) between different nodes (genes). In order to extract a graph from this network it must first be reduced. This is achieved by transforming every pair of inhibition (h) and excitation (e) connections into a single link with the difference between both ($e - h$). Connections whose weight is less than or equal to zero are ignored.

The graph is constructed in a top-down fashion, starting from the output node(s). Recursively, the inputs of each protein in the circuit will be added until only the extra proteins (inputs) are left aside. The order of inclusion in the graph is defined by the corresponding connection weight (see Eq. 2), that is, the strongest connections (with higher binding strength) will be added first. Finally, to avoid recursion, if a node has input connections from others already in the circuit, those are discarded.

When only one output is necessary each product is tested as the graph output (a graph is built and tested for each product). If one wants N outputs from the graph, then the first N products are used as outputs and the graph is built from these in the same fashion (there are no connections amongst the outputs since they do not regulate). Note that the initial individuals usually have fewer proteins and products. If there are not enough products in a network, 0 will be returned for the missing outputs.

4.3 Mapping Proteins to Functions

A mapping is needed to translate nodes (i.e., genes/proteins) to functions. The terminals are mapped sequentially using the input proteins (disregard of their signature). For the remaining nodes we use the gene-protein correspondence.

The protein's signature is used to obtain the function by a majority vote process (each protein codifies a function). As an example, to code the function set `{ +, -, *, / }` only two bits are necessary. The protein's signature is split into sixteen two-bit chunks. Then we obtain the function set index (two bits) by applying the majority vote rule over each bit (two positions) of the sixteen chunks (since the number of chunks is even, in case of a tie the result holds 1). If the function set length is not a power of two, it is either used circularly, or dummy functions may be added.

It is possible for some node to be included in the graph, while not having any input connection available. In this case the proteins' signature is mapped to an integer and wrapped by the hyperbolic tangent function (producing one of the constants -1.0 or 1.0).

5. PROBLEMS

In this section we present the problems used to assess the capabilities of this model. First, we show that the approach is capable of evolving a controller for the inverted pendulum (where only one product is necessary). Second, we test the evolution of graphs with multiple outputs in the context of symbolic expression for Artificial Art.

5.1 The Inverted Pendulum

In this problem there is a cart with an inverted pendulum, the pole, in a finite length 1D track. The goal is to keep the pole balanced and the cart inside the track limits by successively applying a constant force in either direction (left or right) [34].

The function set used was `{ +, -, *, / }`. The terminal set is composed by the position and velocity of the cart, as well as the angle and angular velocity of the pole. Using

these building-blocks a controller is constructed whose output is translated into a push to the left if negative, or to the right if positive.

Each individual was tested over 10 randomised trials, returning the mean fitness of the set. A trial ends when either the cart goes off the track boundaries ($\pm 2.4m$) or the pole falls below 12°. The individual fitness of each trial is defined as in [10]:

$$F(x) = \frac{120000}{\text{number\_of\_successful\_steps}}$$

The evolutionary run terminates when the an individual is found that successfully balances the pole for 120000 steps for each trial, or the maximum number of evaluations is reached. After the evolutionary process the controller is tested for generalisation over $5^4 = 625$ trials (the four input variables, combined using five different rates for each {0.05, 0.275, 0.5, 0.725, 0.95}), during 1000 steps, as described in [34].

5.2 Artificial Art

Evolutionary Art is a growing research field which gathers nowadays researchers from the most varied areas. One of the favourite techniques used to generate evolutionary artifacts is symbolic expression [35]. In the context of GP, it consists in evolving trees (or graphs) that generate for each pixel a colour value which is a function of its coordinates. It is not a goal of the present work to fully describe and improve on the state of art of this field, but rather to provide a proof-of-concept of the use of the model for generating multiple outputs.

The function set used varies amongst researchers, depending on the specific approach/goals. The terminal set is commonly composed of $\{x, y\}$, although variations can be found (for instance, one can use also the distance to the centre, or information from a target image). Fitness can be interactive or automated, both presenting advantages and disadvantages [35].

In this work we intend to test the applicability of the new computational model to this type of problem, by evolving coloured images. The function set that we used was $\{+, -, *, /, sin, cos, sinh, cosh\}$, and the available inputs are the coordinates (x, y), and the distance to the centre r. The colour space used was RGB, so each graph will have three outputs (one for each colour dimension). Fitness automation was not implemented.

Every round the population is displayed on a grid, ordered by fitness score. The run starts with equal fitness for each individual and each time the user chooses one (a mouse click on the interface), its score is increased.

6. EXPERIMENTAL SETUP

The experimental setup for both applications described in the previous section is detailed in Table 1. In the inverted pendulum case the experiment was repeated 50 times, for comparability with results on the literature (as the evaluation is interactive for the artificial artist, a number of runs is not specified). The runs for evolving images lasted from 40 to 60 iterations. The evolution strategy used was an ES-(250+250), while for the artist the strategy used was ES-(25+5). with the initial population generated by a random 32-bit seed submitted to 8 DM-events. The mutation rate used for generating the initial population was 0.02%. For the bit-flip mutation operator a 0.01% rate was applied. In order to allow variable length genomes (aimed at improving efficiency [11]), the *transposon* operator was used in conjunction with the *delete* operator, using 64-bit length sections. The crossover operator is not used in any experiment.

Table 1: Experimental Setup.

Problem	Pendulum	Artificial Art
Number of Runs	50	-
Evolution Strategy	(250+250)	(25+5)
Num. Iterations	10^3	40-60
Number of DMs	8	
DM Mutation Rate	0.02	
Mutation Operator Rate	0.01	
Protein Bind Thresh.	16	
Genome Length	Variable	
Operator Type	Transposon	
Operator Length	64	

7. RESULTS

In the following sections we present the results for both applications. Firstly, we present the results for the pole balancing and compare with other approaches in the literature. Secondly, we display distinct images evolved with the artificial artist.

7.1 Inverted Pendulum

The experiments show that the present model is capable of evolving solutions for the single-output problem of balancing an inverted pendulum. Table 2 averages the generalisation results of the 50 runs, indicating the performance of the best solution found, as well as known results from the literature. Although the approaches are not directly comparable, one can see that this model performs as well as other methods [10, 36], displaying better average over the runs and smaller standard deviation. Also, when compared to [11] there is some deterioration, although the fitness functions used in the present work uses less random initialisations.

Table 2: Summary of the generalisation tests for the cart-pole problem, by number of successful trials out of 625 (see Sect. 5.1). Results from the literature are also transcribed.

Reference	N	Best	Mean	Std. Dev.
ARN-GP	50	434	373.8	48.3
[10]	50	422	202.18	110.01
[36]	50	406	203.18	116.05
[11]	50	497	478	12.63

Figure 6 shows the best circuit found. The output is identified by the hexagonal shape while the inputs are squared. The oval nodes correspond to the transcription factors. It is clear the re-use of functions (nodes) that are used simultaneously as inputs to various other nodes.

7.2 Artificial Artist

In order to test the multiple output features of the model we tried to evolve genomes capable of generating interesting images, by outputting the RGB values for each pixel given the input set `{x,y,r}`. In this section we present some of

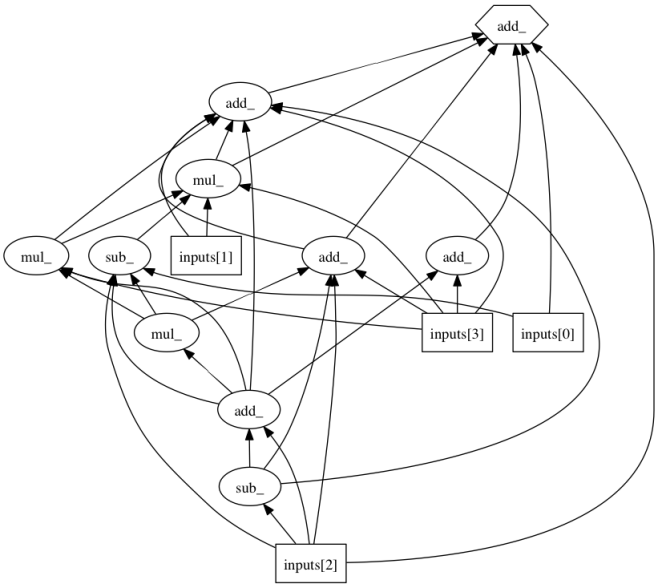

Figure 6: Controller for the best run of the Cart-Pole problem. The output is distinguished by the hexagonal shape, the inputs by the square shape. Oval nodes correspond to the transcription factors. The function set is composed by the arithmetic functions. The input indexes represent, respectively, cart position, pole angle, velocity, and angular velocity.

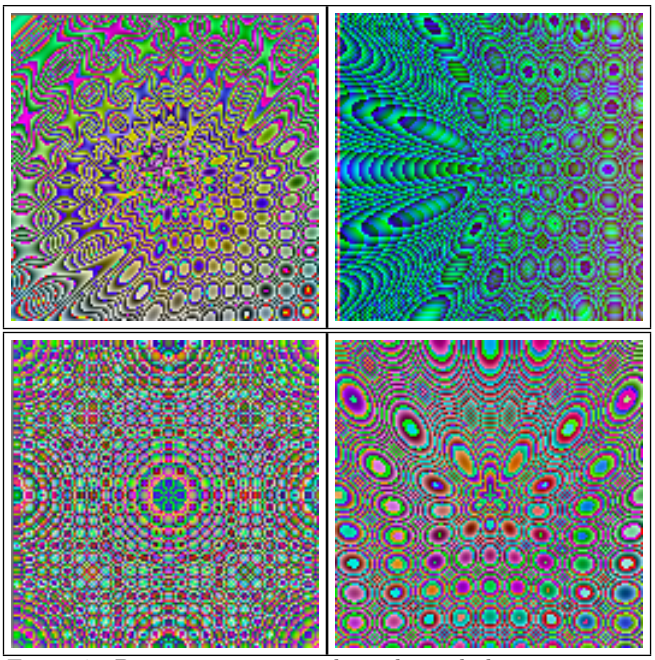

Figure 7: Pattern generation through symbolic expression.

those images, although the corresponding graphs are too big to be displayed in the present format.

In Figure 7 we can see the four most fit individuals of an experiment where the evolutionary process was driven towards images with patterns, composed of circular shapes. A higher resolution version of the picture on the bottom left is presented in Figure 8. The display of patterns and symmetries is clear and one can see some resemblance with fractal images.

Driving the experiment to less complex, but more colourful images we obtained individuals like those presented in Figure 9. A higher resolution version of the picture on the bottom left is presented in Figure 10. Despite the simplicity of this image some symmetries and patterns are also present.

These experiments demonstrate the use of the present model to generate parallel outputs. Although only interactive evolution was used, we have shown the ability of the approach to generate solutions with different properties. It is our believe that it provides a proof-of-concept for future applications of this hybrid.

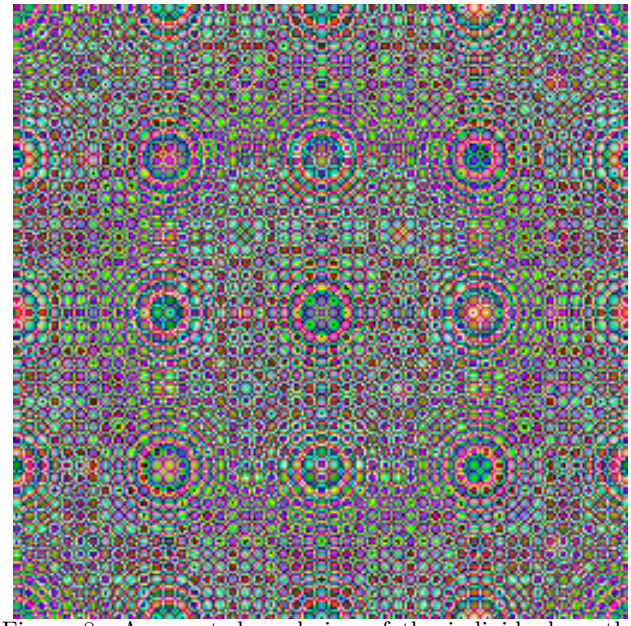

Figure 8: Augmented rendering of the individual on the bottom left of Fig. 7.

8. CONCLUSION

A fusion between the ReNCoDe and the extended ARN models was presented, aimed at solving a different class of problems. This method allows the extraction of executable graphs with one or more outputs from an artificial regulatory network. Moreover, the algorithm for the graph extraction is more clear with the ordering based on the protein binding strengths.

We have tested the hybrid in a typical GP benchmark problem - the inverted pendulum - showing that the approach is competitive with others in the literature, including the original ReNCoDe. In the domain of multiple output

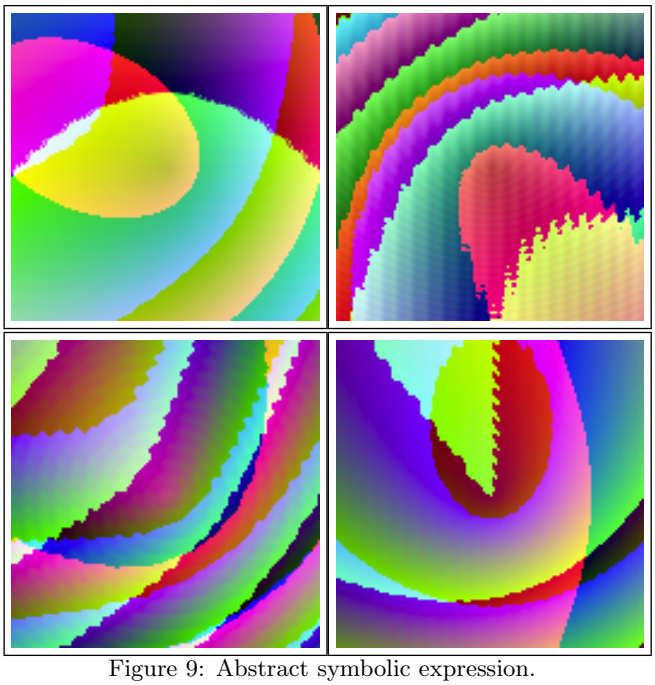

Figure 9: Abstract symbolic expression.

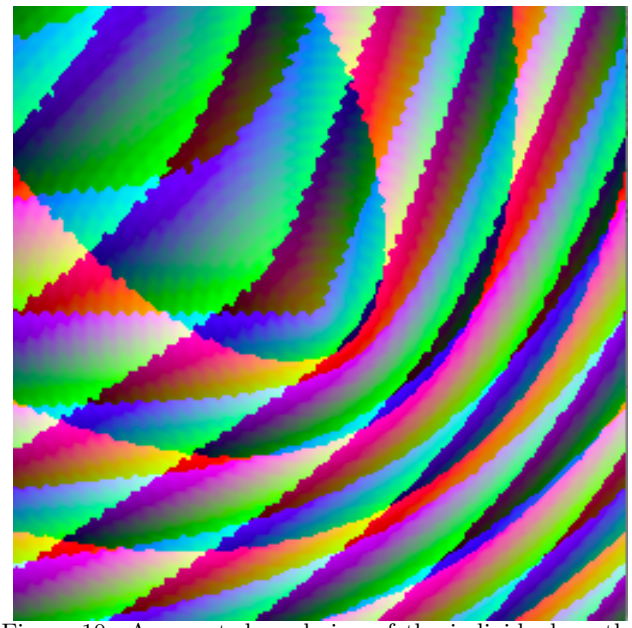

Figure 10: Augmented rendering of the individual on the bottom left of Fig. 7

applications, we provided a proof-of-concept for the methodology by testing it in an artificial art context. The evolved images display complex repetitive patterns and symmetries, as well as simpler forms, more colourful, also displaying symmetries and repetition with variation.

Future work will test the model's ability to evolve solutions for complex binary problems, such as the full adder and binary multipliers of different input sizes.

Acknowledgements

The work of the first author was partially funded by Fundação para a Ciência e Tecnologia, grant ref. SFRH / BD / 69106 / 2010.

9. REFERENCES

[1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer Verlag, 2003.

[2] M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf, "Open issues in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, pp. 339–363, 2010.

[3] W. Banzhaf, G. Beslon, S. Christensen, J. Foster, F. Képès, V. Lefort, J. Miller, M. Radman, and J. Ramsden, "From artificial evolution to computational evolution: a research agenda," *Nature Reviews Genetics*, vol. 7, no. 9, pp. 729–735, 2006.

[4] M. O'Neill and C. Ryan, *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language*. Genetic programming, Kluwer Academic Publishers, 2003.

[5] J. F. Miller, *Cartesian Genetic Programming*. Natural Computing Series, Springer, 2011.

[6] C. Ferreira, *Gene Expression Programming (2nd Edition)*. Springer, 2006.

[7] M. A. Lones and A. M. Tyrrell, "Biomimetic representation with genetic programming enzyme," *Genetic Programming and Evolvable Machines*, vol. 3, pp. 193–217, June 2002.

[8] W. Banzhaf, "Artificial regulatory networks and genetic programming," in *Genetic Programming Theory and Practice* (R. L. Riolo and B. Worzel, eds.), ch. 4, pp. 43–62, Kluwer, 2003.

[9] P. Dwight Kuo, W. Banzhaf, and A. Leier, "Network topology and the evolution of dynamics in an artificial genetic regulatory network model created by whole genome duplication and divergence.," *Bio Systems*, vol. 85, no. 3, pp. 177–200, 2006.

[10] M. Nicolau, M. Schoenauer, and W. Banzhaf, "Evolving Genes to Balance a Pole," in *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010* (A. I. and others Esparcia-Alcazar, ed.), vol. 6021 of *LNCS*, (Istanbul), pp. 196–207, Springer, 2010.

[11] R. Lopes and E. Costa, "The regulatory computational device," *Genetic Programming and Evolvable Machines*, vol. 13, no. 3, pp. 339–375, 2012.

[12] H. Bolouri, *Computational modeling of gene regulatory networks: a primer*. Imperial College Press, 2008.

[13] H. de Jong, "Modeling and simulation of genetic regulatory systems: a litterature review," *Journal of Computational Biology*, vol. 9, no. 1, pp. 67–103, 2002.

[14] Â. Gonçalves and E. Costa, "A model for an heterogeneous gene regulatory network," in *Handbook of Research on Computational Methodologies in Gene Regulatory Networks* (S. Das, D. Caragea, S. M. Welch, and W. H. Hsu, eds.), ch. Chapter 12, IGI Global, 2009.

[15] M. Hecker, S. Lambeck, S. Toefler, E. van Soemren, and R. Guthke, "Gene regulatory network inference: data integration in dynamic models - a review," *BioSystems*, vol. 96, no. 1, pp. 86–103, 2009.

[16] D. Marbach, C. Mattiussi, and D. Floreano, "Bio-mimetic evolutionary reverse engineering of gene regulatory networks," in *5th European Conference on Evolutionary Computation,Machine Learning and Data Mining in Bioinformatics, EvoBIO2007* (E. Marchiori, J. H. Moore, and J. C. Rajapakse, eds.), vol. 4447 of *LNCS*, pp. 155–165, Springer, 2007.

[17] E. Sakamoto and H. Iba, "Inferring a system of differential equations for a gene regulatory network by using genetic programming," in *Proceedings of the Congress on Evolutionary Computation*, pp. 720–726, IEEE Press, 2001.

[18] F. Streichert, H. Planatscher, C. Spieth, H. Ulmer, and A. Zell, "Comparing genetic programming and evolution strategies on inferring gene regulatory networks," in *Proceeding of Genetic and Evolutionary computation Conference (GECCO2004)*, vol. 3102 of *LNCS*, pp. 471–480, 2004.

[19] N. Noman and H. Iba, "Inferring gene regulatory networks using differential evolution with local search heuristics," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 4, pp. 634 –647, oct.-dec. 2007.

[20] T. Reil, "Dynamics of gene expression in an artifical genome - implications for biological and artificial ontogeny," in *Proceedings of the 5th European conference on Artificial Life* (D. Floreano, J.-D. Nicoud, and F. Mondada, eds.), pp. 457–466, Springer, 1999.

[21] S. Kauffman, *The origins of order: self-organization and selection of evolution*. Oxford University Press, 1993.

[22] H. de Jong, J.-L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann, "Hybrid modeling and simulation of genetic regulatory networks," in *Proceedings of the 6th international conference on Hybrid systems: computation and control*, HSCC'03, (Berlin, Heidelberg), pp. 267–282, Springer-Verlag, 2003.

[23] P. Eggenberger, "Evolving morphologies of simulated 3D organisms based on differential gene expression," in *Fourth European Conference of Artificial Life* (P. Husbands and I. Harvey, eds.), pp. 205–213, MIT Press, 1997.

[24] D. Roggen, D. Federici, and D. Floreano, "Evolutionary morphogenesis for multi-cellular systems," *Genetic Programming and Evolvable Machines*, vol. 8, pp. 61–96, Mar. 2007.

[25] M. Joachimczak and B. Wrobel, "Evo-devo in silico: a model of a gene network regulating multicellular development in 3d space with artificial physics," in *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems* (S. Bullock, J. Noble, R. Watson, and M. A. Bedau, eds.), pp. 297–304, MIT Press, 2008.

[26] J. Knabe, M. Schilstra, and C. Nehaviv, "Evolution and morphogenesis of differential multicellular organisms: autonomously generated diffusion gradients for positional information," in *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems* (S. Bullock, J. Noble, R. Watson, and M. A. Bedau, eds.), pp. 321–328, MIT Press, 2008.

[27] M. Kessler, *Analysis of the dynamics of a GRN-based Evo-devo system*. PhD thesis, University of Zurich, September 2009.

[28] J. Bongard, "Evolving modular genetic regulatory networks," in *IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1872–1877, IEEE Press, 2002.

[29] T. Quick, C. Nehaviv, K. Dautenhahn, and G. Roberts, "Evolving embodied genetic regulatory network-driven control systems," in *Proceedings of the European Conference on Artificial Life (ECAL 2003)* (W. Banzhaf, ed.), vol. 2801 of *Lecture Notes in Artificial Intelligence*, pp. 266–277, 2003.

[30] P. Kennedy and T. Osborn, "A model of gene expression and regulation in an artificial cellular organism," *Complex Systems*, vol. 13, no. 1, pp. 33–59, 2001.

[31] S. A. Teichmann and M. M. Babu, "Gene regulatory network growth by duplication.," *Nature Genetics*, vol. 36, pp. 492–6, May 2004.

[32] P. Kuo, A. Leier, and W. Banzhaf, "Evolving dynamics in an artificial regulatory network model," *Lecture Notes in Computer Science*, pp. 571–580, 2004.

[33] M. Nicolau, M. O'Neill, and A. Brabazon, "Applying Genetic Regulatory Networks to Index Trading," *Parallel Problem Solving from Nature-PPSN XII*, pp. 428–437, 2012.

[34] D. Whitley, S. Dominic, R. Das, and C. W. Anderson, "Genetic reinforcement learning for neurocontrol problems," *Machine Learning*, vol. 13, no. 2, pp. 259–284, 1993.

[35] P. Machado, J. J. Romero, and A. Carballal, *Evolutionary and Biologically Inspired Music, Sound, Art and Design*. First International Conference, EvoMUSART 2012, Málaga, Spain, April 11-13, 2012, Proceedings, Springer-Verlag New York Incorporated, Mar. 2012.

[36] E. Murphy, M. Nicolau, E. Hemberg, M. O'Neill, and A. Brabazon, "Differential Gene Expression with Tree-Adjunct Grammars," *Parallel Problem Solving from Nature-PPSN XII*, pp. 377–386, 2012.