Differential Gene Expression with Tree-Adjunct Grammars

Eoin Murphy, Miguel Nicolau, Erik Hemberg, Michael O'Neill, and Anthony Brabazon

Natural Computing Research and Applications Group,
Univeristy College Dublin, Ireland
{eoin.murphy,miguel.nicolau,erik.hemberg,m.oneill,anthony.brabazon}@ucd.ie

Abstract. A novel extension of an existing artificial Gene Regulatory Network model is introduced, combining the dynamic adaptive nature of this model with the generative power of grammars. The use of grammars enables the model to produce more varied phenotypes, allowing its application to a wider range of problems. The performance and generalisation ability of the model on the inverted-pendulum problem, using a range of different grammars, is compared against the existing model.

1 Introduction

Recently in the field of Evolutionary Computation there has been an increase of interest in developmental biology and how it affects the evolutionary models currently in use. It is interesting to investigate whether these developmental systems can be applied to improve existing evolutionary approaches, as developmental processes play such an important role in nature.

To date, much of the work that has been done with developmental systems in the field of Genetic Programming (GP) has been ontogenetic in nature. That is to say, concerning with the growth or morphogenesis of individuals, by means of interaction with the environment [12, 5, 4]. These developmental systems derive the phenotype from the genotype, evaluating this phenotype, before undergoing some morphogenesis defined within the phenotype itself. This process continues, creating more complex phenotypes. The organism morphologies in nature on which these systems are modeled on, however, are themselves the result of an underlying regulatory process, known as gene regulatory networks (GRNs).

GRNs, which are at the core of developmental biology allow for *"differential gene expression from the same nuclear repertoire"* [3]. This is partly achieved by including feedback loops and being directly influenced by the environment. An outline of a new genetic representation for GP using GRNs was given by Banzhaf [1], exploring the dynamics of such a system. This new representation has since been extended, providing methods for encoding the state of the environment into the GRN, as well as extracting a signal from the GRN [10]. However, even in its extended form, the phenotype produced is a series of signals between 0.0 and 1.0, making this representation difficult to apply to many different problems.

This study is concerned with applying the generative power of grammars to the model, and investigating the effects of this complexification of the mapping process. Making use of grammars to generate phenotypes enables the model to be applied to many different problems. Tree-Adjunct Grammatical Evolution (TAGE) [9], a variant of the popular grammar-based form of GP, Grammatical Evolution (GE), is extended to include the GRN model. TAGE is ideal due to a unique property of the grammar type it employs, Tree-Adjoining Grammars (TAGs) [6], which produce valid individuals at every stage of derivation.

The following section gives introductions to TAGE and the GRN model. Section 3 presents the extension of the GRN model into the TAGE algorithm, followed by a description of the pole-balancing problem in section 4. The experiments performed and results obtained are presented with some discussion in section 5. The study concludes in section 6, outlining some future work.

2 Background

2.1 Tree-Adjunct Grammatical Evolution

TAGE is a grammar-based form of GP, combining aspects of Darwinian natural selection, genetics and molecular biology with the representational power of grammar formalisms [11, 9]. TAGE uses a representation consisting of a TAG and a chromosome. A TAG is defined by a quintuple (T, N, S, I, A) where: T is a finite set of terminal symbols; N is a finite set of non-terminal (NT) symbols: $T \cap N = \emptyset$; S is the start symbol: $S \in N$; I is a finite set of finite trees called *initial trees*; and A is a finite set of finite trees called *auxiliary trees*.

Initial trees represent the minimal non-recursive structures produced by the grammar, i.e., they contain no repeated NT symbols. Inversely, auxiliary trees of type `X` represent the minimal recursive structures, which allow recursion upon the NT `X`. The union of initial trees and auxiliary trees forms the set of *elementary trees*, E; where $I \cap A = \emptyset$ and $I \cup A = E$.

During derivation, the adjunction composition operation make use of codons to join elementary trees together. When using TAGs, at each stage of derivation, before and after each adjunction operation, valid phenotypes can be extracted. Due to space constraints, a full description of TAGE has been omitted, but can be found in [9]. In addition, the grammars used throughout this study are presented in Fig. 4 as context-free grammars (CFG) in an effort to save space. TAGE can transform between CFGs and equivalent TAGs.

2.2 Artificial Gene Regulatory Networks

The GRN model [1] used in this study, which has been extended to enable input and output [10], consists of three components: a genome, genes, and proteins. The model mimics the biological interaction of proteins with a cell's genes. By binding at *regulatory sites*, certain proteins can regulate the expression of genes, and hence, the production of additional proteins. Proteins are assigned concentrations in the model, with a total concentration of 1.0 for each type of protein.

The term *concentration*, with regards to this model, represents the proportion of a particular protein to the rest of the proteins of the same type.

Gene. A gene consists of four sections, as shown in Fig. 1. The first two are two 32 bit regulatory sites, the *enhancer* and *inhibitor* sites. These two sites affect a gene's expression positively or negatively, respectively. A 32 bit *promoter site*, which follows these regulatory sites, is of the form `XYZ01010101` where `XYZ` is an arbitrary 24 bit sequence and the final eight bits define the gene's type or signature. The specific eight bit signature of a promoter is used to identify genes along the genome. Following the promoter is a 160 bit region, which encodes the protein. These 160 bits are subdivided into five 32 bit sections. A majority vote is performed at each bit position across the five sections to determine the 32 bits making up the protein signature, as shown in Fig. 2.

Gene/Protein Types and Input/Output. Genes are split into two groups in this model, *Transcription Factors* (TF-genes) and *Products* (P-genes). A gene's type is dependent upon its promoter's signature, in this case `XYZ00000000` is used to identify TF-genes and `XYZ11111111` to identify P-genes. TF-genes produce TF-proteins which can bind to regulatory sites and affect gene expression. P-genes produce P-proteins which are are used solely to extract output from the model, and as such, are prevented from binding and affecting regulation. The sum of concentrations for each protein type must add to 1.0. Input into the model is achieved by injecting specific concentrations of *free* TF-proteins into the model. Input values are encoded into the proteins' concentration levels. The concentration of free proteins remains static unless new inputs are injected.

Regulation. How much each protein affects a gene's expression by binding at that gene's regulatory sites (enhancer and inhibitor) is calculated by taking the XOR of the protein's signature with that of the regulatory site. The number of bits set is the degree of match between the two, i.e., the number of complementary bits between them. The enhancing, e_i, and inhibiting, h_i, signals for the expression of gene g_i are calculated as follows:

$$e_i, h_i = \frac{1}{N}\sum_{j=1}^{N} c_j e^{\beta(u_j - u_{max}))} \ , \tag{1}$$

where N is the total number of TF-proteins, c_j is the concentration of protein j, u_j is the number of complementary bits between the regulatory site and protein j, u_{max} is the maximum number of complementary bits observed in the system, and β is a positive scaling factor. A value of 1.0 was used for scaling factors β and δ across all experiments.

The expression rate of g_i (the production of p_i) at time $t+1$ is given as:

$$\frac{dc_i}{dt} = \delta(e_i - h_i)c_i \ , \tag{2}$$

where δ is a scaling factor. Once all produced TF-protein concentrations have been updated, these concentrations are scaled such that when summed with the free TF-protein concentrations they add to 1.0.

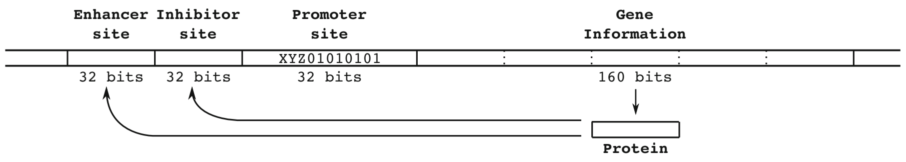

Fig. 1. A gene on the genome split into its different sections

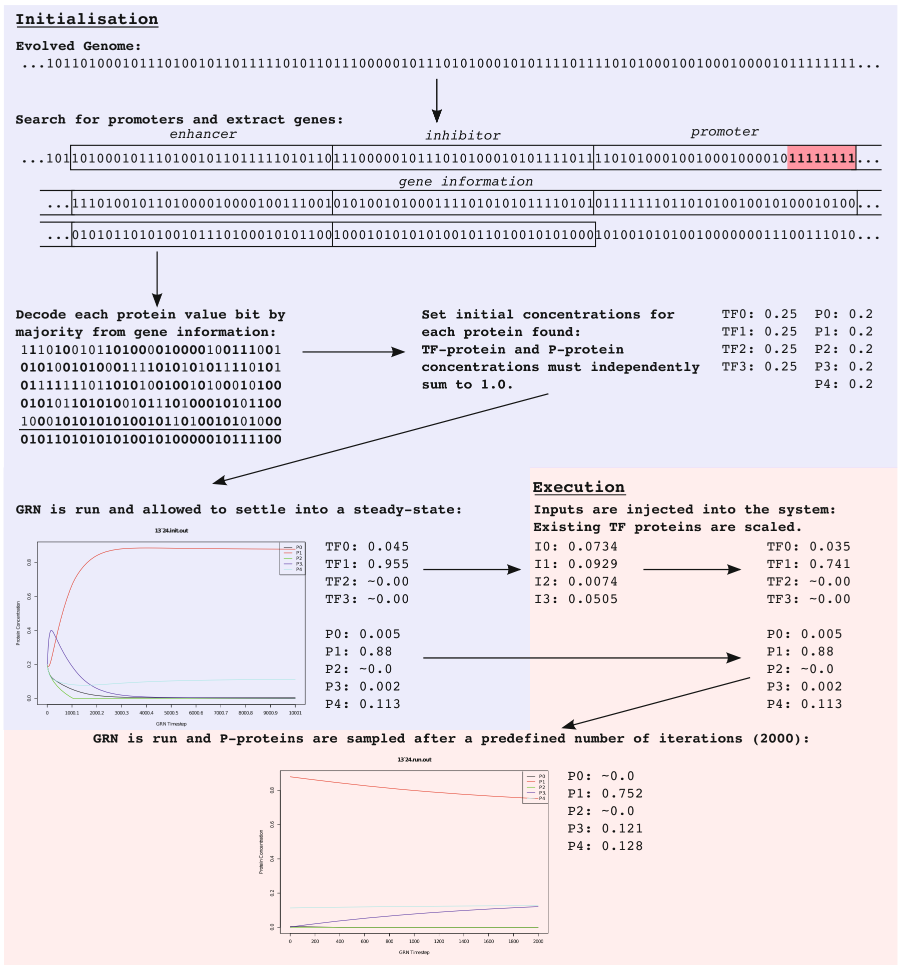

Fig. 2. An illustrative example of the GRN model. The genome is scanned for promoter sites and genes are extracted. Each gene's protein signature is calculated and each protein is given an initial concentration. The system is run for a number of iterations to become stable. The environment state is then read, encoded as TF-protein concentrations and injected into the system, scaling the existing TF-protein concentrations. The system is run for a number of iterations before extracting P-proteins to be used for mapping. This process of encoding and injecting inputs, iterating and extracting outputs can continue indefinitely.

A similar formula is used for the expression rate of P-genes. P-protein concentrations are normalised separately from the TF-protein concentrations to sum to 1.0.

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

3 Combining Artificial GRNs and Grammars

In order to enable the use of GRNs within the TAGE algorithm, the TAGE pipeline must be extended. The GRN model is embedded at the start of the mapping process. This enables easy access to the evolved genome in order to search for TF-genes and P-genes to construct the GRN, as well as providing access to the traditional mapping process to remap the phenotype each time the fitness function/environment changes. In order to allow for this, a feedback loop is required from the fitness function to the mapper.

To evaluate an individual, its genome must first be mapped. The mapping call initiates a search of the individual's genome for genes and a GRN is constructed. This GRN is allowed to run, without free TF-proteins (inputs), for 10000 iterations in an effort to allow the network to settle into a steady state. This may stop prematurely if a steady state within the GRN is detected earlier.

When the fitness function is called to evaluate the individual, the initial state of the environment, encoded as concentrations of free TF-proteins, is passed to the mapper and injected into the GRN, scaling the existing produced TF-proteins as necessary. The GRN is then run for a predefined number of iterations before the P-proteins and their concentrations are used to create a string of integer codon values. These codons are then used by the traditional mapping process to produce a valid phenotype. One of the major advantages of using TAGs is that regardless of how many codon values are available to the mapper, a valid phenotype can always be produced.

3.1 Using Product-Proteins as Mapping Inputs

There are many different methods of interpreting P-proteins and their concentration levels to produce codon values for mapping. In this study, four such methods are examined, two for use with a binary grammar and two for use with grammars of any arity. The first two methods are chosen to simulate the approaches taken in [10] while still retaining the use of a direct mapping grammar. These two methods make use of a single P-protein to produce a codon value, whereas the remaining two methods use all available P-proteins to produce a chromosome of codon values, enabling mapping with grammars of much higher complexity.

Concentration Value. The P-protein's concentration is examined. If it is found to be greater than 0.5, a codon value of `1` is used. Otherwise, a codon value of `0` is used. Using Fig. 2 as an example, After the model has iterated, `P0` has a concentration of ~ 0.0 producing a codon value of `0`. This results in the tree at index `0` from the grammar being chosen.

Concentration Tendency. The change in concentration of a P-protein from input injection until the final iteration. If the magnitude of this change is greater than some *threshold*, a codon value of 0 or 1 is used depending on the sign of that change, positive or negative respectively. Otherwise, the codon value chosen previously will be reused. Taking Fig. 2 as an example, after the inputs have been injected, `P0` has concentration of 0.05, and a concentration of ~ 0.0 after iteration `2000`. As $|-0.05|$ is both greater than the *threshold* of 1^{-10} and is negative, a codon value of `1` is used, choosing the initial tree from index `1` in the grammar.

Sort by Concentration. The set of P-proteins is sorted in descending order by concentration level, as was suggested in [1]. The P-protein's 32 bit signatures are then used to represent integer codon values, with the most concentrated P-protein being the left-most codon. In Fig. 2, the P-proteins when sorted by concentration are `P1, P4, P3, P0, P2`, using their signatures as codons in this order, an initial tree is chosen from the grammar (one codon) and two adjunction operations are performed (two codons each).

Sort by Concentration Tendency. Similar to the *Sort by Concentrated* method above, however the P-proteins are sorted by means of the signed magnitude of the tendency of their concentration values. In this case, the P-proteins are sorted in the order `P3, P4, P2, P0, P1` as `P3` has the largest increase of 0.119 and `P1` with the smallest of -0.128. The protein signatures are used as codon values in this order with the grammar to create a phenotype.

4 The Inverted Pendulum (Pole-Balancing) Problem

The problem examined throughout the course of this study is the pole-balancing problem [2, 13], a classic dynamic control problem that has recently been examined in the study of applying GRNs to evolutionary computation [10, 8]. The description below is based on the problem setup used by Nicolau et al. [10].

The problem consists of simulating a cart which can move along a finite two dimensional track. A rigid pole is hinged to the centre of the cart. Forces may be applied to the cart in either direction, causing the cart and the pole to accelerate either left or right. The aim of the problem is to prevent the angle created between the pole and the vertical from becoming greater than some threshold, and keeping the cart within the bounds of the track. The problem model consists of four state variables:

$x \in [-2.4, +2.4]m$ the cart position, relative to the centre of the track;
$\dot{x} \in [-1.0, 1.0]m/s$ the velocity of the cart;
$\theta \in [-12, 12]^\circ$ the angle of the pole with the vertical;
$\dot{\theta} \in [-1.5, 1.5]^\circ/s$ the angular velocity of the pole.

The (friction-less) physical simulation of the cart-pole model is governed by the following non-linear differential equations:

$$\ddot{\theta}_t = \frac{g \sin\theta_t - \cos\theta_t \left[\frac{F_t + ml\dot{\theta}_t^2 \sin\theta_t}{m_c + m}\right]}{l\left[\frac{4}{3} - \frac{m\cos^2\theta_t}{m_c + m}\right]} \qquad \ddot{x}_t = \frac{F_t + ml\left[\dot{\theta}_t^2 \sin\theta_t - \ddot{\theta}_t \cos\theta_t\right]}{m_c + m}, \quad (4)$$

where $g = -9.8m/s^2$, acceleration due to gravity, $m_c = 1,0kg$, the mass of cart, $m = 0.1kg$, the mass of pole, $l = 0.5m$, the half-pole length, $F_t = \alpha \cdot 10N$ where $\alpha \in [-1.0, 1.0]$, the force applied to the cart's center of mass at time t.

These variables are encoded as free TF-proteins for the GRN model by assigning a unique signature to each variable, with each variable's value normalised in the range $[0.0, 0.1]$ taking up a max concentration of 0.4 in the GRN model. The signatures of each variable are:

x: `00000000000000000000000000000000` θ: `11111111111111111111111111111111`
$\dot{x}$: `11111111111111110000000000000000` $\dot{\theta}$: `00000000000000001111111111111111`

These free TF-proteins are injected into the stabilised GRN and the existing produced TF-proteins are normalised as required. The GRN is then iterated 2000 times, corresponding to $0.2s$ of simulated time for the cart-pole model. After which, the P-proteins are mapped and the phenotype is evaluated resulting in a scaling co-efficient, α, for the force. The value of α is clamped between -1.0 and 1.0 inclusive. The scaled force is applied to the equations of motion and the state variables are updated, re-encoded, and fed back into the GRN. This process continues until the maximum number of time steps is reached, or the cart-pole model enters a failure state, i.e., $-2.4 > x > 2.4$ or $-12° > \theta > 12°$. Fitness is calculated by:

$$F(x) = \frac{120000}{successful\ time\ steps} - 1 \ . \tag{5}$$

5 Experiments

This study is concerned with the application of grammars to the artificial GRN model. Several experiments are presented to help to examine this effect. The grammars used throughout this study are listed in Fig. 4, with the general evolutionary parameters of the system used in Tab. 1. For each run, the pole-cart model is initialised with a random state at the start of each generation.

Table 1. GE parameters adopted for each of the benchmark problems

Parameter	Value
Generations	50
Population Size	250
Initialisation	Random
Chromosome Size	128 (4096 bits)
Replacement Strategy	Generational
Elitism	25 Individuals
Selection Operation	Tournament
Tournament Size	3
Bit Mutation Prob	0.005

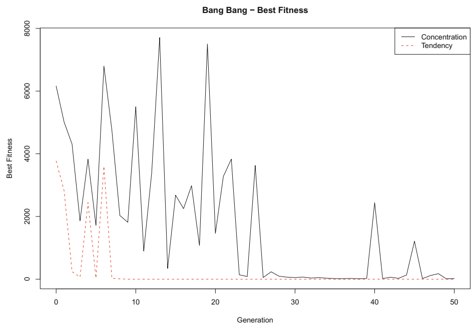

Fig. 3. Mean best fitness plot

```
<power>  ::= 1.0 0.0 - | 1.0
```

(a) Direct mapping grammar

```
<power>  ::= <const> 0.0 <op>
<op>    ::= + | -
<const> ::= 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
          0.6 | 0.7 | 0.8 | 0.9 | 1.0
```

(b) Discrete digits grammar

```
<power>  ::= <const> 0.0 <op>
<op>    ::= + | -
<const> ::= 0.<digits>
<digits> ::= <digit><digits> | <digit>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 |
          6 | 7 | 8 | 9
```

(c) Continuous digits grammar

```
<power> ::= <expr> <expr> <op>
<expr>  ::= <expr> <expr> <op> | <const>
<op>    ::= + | - | * | /
<const> ::= 0.<digit>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 |
          6 | 7 | 8 | 9
```

(d) Symbolic regression grammar

Fig. 4. The grammars used, in reverse polish notation. Note that (c) generates the range $(-0.9, 0.9)$ and the values of (d) are clamped to $[-1.0, 1.0]$ as previously.

In order to determine whether the system used by this study is comparable to the work done previously by Nicolau et al. [10], 50 independent runs of the simple grammar using the first and second output methods (see Section 3.1) are performed, using only a single codon value, and producing either `-1` or `+1`. Following this, assessing the effect of grammar complexity on the system, three different grammars of increasing complexity are examined over 50 runs using the remaining two P-protein mapping approaches.

5.1 Analysis of Results

The mean best fitness plots of the direct mapping grammar experiments are presented in Fig. 3. From the plot, it appears that the P-protein concentration tendency approach performs better, managing to find solutions in far fewer generations than using the protein's concentration value. The concentration tendency approach also manages to find more solutions overall than the value approach.

While these plots are similar in trend to those presented in [10], showing a similarity in performance, Fig. 3 has a higher variance. This variance can be accounted for by the choice of selection methods, generational vs steady-state [10]. Similar plots are presented for the three other grammars in Fig. 5. From these plots, it appears that using the concentration tendency approach is better than using the protein concentration itself.

Generalisation Results. The generalisation test proposed by Whitley et al. [13] is used in this study to examine robustness. After each run, the best individual is tested for 1000 time steps on 625 test cases. For each of the problem's four state variables, a set of five values is calculated, with these values normalised to 0.05, 0.275, 0.725 and 0.95 of the each variable's range. The test cases are obtained from combining these values. However, only `457` cases are solvable [10].

Given how quickly solutions are found by the tendency approach, as seen in Fig. 5, it is not surprising that this approach does not perform as well in terms of generalisation. Tab. 2 shows that while finding fewer successful solutions, the concentration value approach performs better in the generalisation tests.

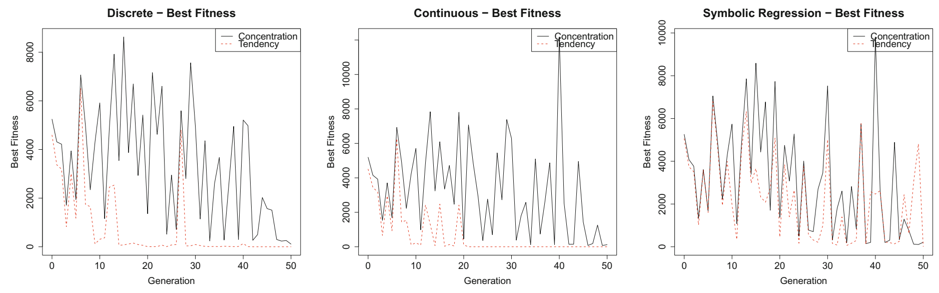

Fig. 5. Mean best fitness plots for the three more complex grammars

Table 2. Generalisation test results: 1000 time steps for 625 test cases

Approach	Best	Worst	Median	Mean	Std. Dev.	Suc. (50)
Best Product - Concentration	406	0	203	203.18	116.05	47
Best Single Output - Tendency	137	0	52	57.52	34.43	50
Discrete - Concentration	355	0	110	120.7	111.76	36
Discrete - Tendency	240	20	87	88.68	45.13	50
Continuous - Concentration	390	0	162	155.48	118.46	40
Continuous - Tendency	200	10	51	61.62	36.35	50
Sym. Reg. - Concentration	356	0	111	128.44	110.21	39
Sym. Reg. - Tendency	208	0	69	78.82	51.75	43

This could be due to the fact that perfect individuals which use this approach appear much later in the run, with the population having being exposed to more instances of the problem than those using the tendency approach. Allow evolution to continue for the whole run rather than stopping once a solution to any instance of the problem is found might help counteract this.

6 Conclusions

The objective of this study was to examine the effect of integrating grammars and an artificial GRN model, the motivation for which is two fold. Firstly, advances in developmental biology have increased our understanding of developmental processes and how they affect the natural evolution; this knowledge is slowly filtering into EC fields such as GP. Secondly, while artificial GRNs have been applied to the field of GP previously [10], the phenotypes have been limited to simple signals. By exploiting the generative power of grammars, these systems can be extended to have more varied phenotypic products, allowing their application to a broader range of problem domains.

Addressing this, an artificial GRN model was adapted into the TAGE algorithm. Different methods of extracting output from the GRN model for mapping using a grammar were examined. The results obtained show that the inclusion of a grammar is not detrimental to performance of the GRN in the simplest case, reproducing results previously published in the literature [10], with more complex grammars highlighting the system's ability to produce and operate effectively with a more complex genotype-phenotype mapping.

Future work includes the further study of the model, in particular the time complexity of the algorithm, i.e., the computational cost of initialising the GRN and processing the output proteins for mapping. The method will also be compared against other approaches to the pole balancing problem, such as [7], as well as harder instances of the problem. In addition, the system will be applied to to other dynamic control problems as the uniquely inherent dynamism provided by the GRN could prove to be beneficial.

Acknowledgements. This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868.

References

[1] Banzhaf, W.: Artificial regulatory networks and genetic programming. In: Genetic Programming Theory and Practice, ch.4, pp. 43–62. Kluwer (2003)

[2] Barto, A.G., Sutton, R.S., Anderson, C.W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, & Cybernetics, 834–846 (September- October 1983)

[3] Gilbert, S.F.: Developmental Biology, 8th edn. Sinauer Associates Inc. (2006)

[4] Harding, S., Miller, J.F., Banzhaf, W.: SMCGP2: self modifying cartesian genetic programming in two dimensions. In: GECCO 2011: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, July 12-16, pp. 1491–1498. ACM, Dublin (2011)

[5] Hoang, T.H., Essam, D., McKay, R.I.B., Hoai, N.X.: Developmental evaluation in genetic programming: The TAG-based frame work. International Journal of Knowledge-Based and Intelligent Engineering Systems 12(1), 69–82 (2008)

[6] Joshi, A., Schabes, Y.: Tree-Adjoining Grammars. Handbook of Formal Languages, Beyond Words 3, 69–123 (1997)

[7] Khan, M.M., Khan, G.M., Miller, J.F.: Evolution of neural networks using cartesian genetic programming. In: IEEE Congress on Evolutionary Computation (CEC 2010), July 18-23. IEEE Press, Barcelona (2010)

[8] Lopes, R.L., Costa, E.: ReNCoDe: A Regulatory Network Computational Device. In: Silva, S., Foster, J.A., Nicolau, M., Machado, P., Giacobini, M. (eds.) EuroGP 2011. LNCS, vol. 6621, pp. 142–153. Springer, Heidelberg (2011)

[9] Murphy, E., O'Neill, M., Galvan-Lopez, E., Brabazon, A.: Tree-adjunct grammatical evolution. In: 2010 IEEE World Congress on Computational Intelligence, pp. 4449–4456. IEEE Press, Barcelona (2010)

[10] Nicolau, M., Schoenauer, M., Banzhaf, W.: Evolving Genes to Balance a Pole. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 196–207. Springer, Heidelberg (2010)

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

[12] Spector, L., Stoffel, K.: Ontogenetic programming. In: Genetic Programming 1996: Proceedings of the First Annual Conference, pp. 394–399. MIT Press, USA (1996)

[13] Whitley, D., Dominic, S., Das, R., Anderson, C.W.: Genetic reinforcement learning for neurocontrol problems. Machine Learning 13, 259–284 (1993)