Evolving Reactive Controller for a Modular Robot: Benefits of the Property of State-Switching in Fractal Gene Regulatory Networks

Payam Zahadat, Thomas Schmickl, and Karl Crailsheim

Artificial Life Lab of the Department of Zoology,
Universitätsplatz 2,Karl-Franzens University Graz, 8010 Graz, Austria
payam.zahadat@uni-graz.at

Abstract. In this paper, we study Fractal Gene Regulatory Networks (FGRNs) evolved as local controllers for a modular robot in snake topology that reacts adaptively to environment. The task is to have the robot moving in a specific direction until it reaches a randomly placed target-zone and stays there. We point to a characteristic of FGRN model, namely "state-switching property" and demonstrate it as a beneficial property in evolving reactive controllers.

Keywords: Gene regulatory network, reactive controller, modular robot

1 Introduction

A real world task for a robot usually consists of several parts where each part demands for a specific behavior. The robot needs to sense environment and react properly by regulating its behavior. In evolutionary robotics, evolvability for reactive controllers which are able to switch between different behaviors based on the environmental signals seems to be an important issue to consider. This paper demonstrates evolvability of Fractal Gene Regulatory Networks (FGRNs) as decentralized reactive controllers for a modular robot that adaptively reacts to environment. The paper points to a special characteristic of FGRN we will refer to as "state-switching property" as a potential strength of the method making it suitable for evolving reactive controllers.

A modular robot is made up from a number of mechanically coupled modules where each module is typically controlled by its own local controller. A local controller can be a computational Gene Regulatory Network (GRN) that is a network of genes which produce outputs during their interactions. GRN-like systems are previously used to control different modular robots in relatively limited tasks [17, 14, 7]. Each module of the robot is controlled by a single GRN as a control unit. All the GRN units are genetically identical but runs in parallel and behave variously based on the signals receiving from local sensors. Models of GRNs are inspired by interactions happening inside of a biological cell. Unlike the standard Evolutionary Computational (EC) systems, in a GRN system, phenotypes

are indirectly generated from genotypes. A mediatory process of development is required to generate the phenotype through interactions between the genes and between the genes and the environment. Based on the inspiration source of the GRN models the designers anticipate to reach acceptable levels of evolvability for their systems. Many different variants of GRNs are defined by researchers in the field, e.g. [13, 1, 5, 4]. Different GRNs are various in defining the methods of interactions and encoding representations for their genomes which influence evolvability of the system by affecting the shape of the fitness landscape. Apart from the explanatory definition of the interactions during developmental process and representations, a GRN model is eventually a computational network. The model implicitly defines a potentially complex network of nodes and equations that determine the output of each node. Taking a closer look into a particular GRN model might be helpful to get a better understanding of it and identifying the causes of strengths or weaknesses of the model. It may also lead to design new versions of GRNs with improved capabilities.

FGRN [3] is a variant of computational GRNs. It has been successfully evolved for different tasks such as producing patterns [3], controlling single robots [2], motion planning [18], pole-balancing [9], and controling modular robots [17].Dynamics of an FGRN system can be described as several conditional sets of differential equations which are implicitly encoded in the genotype [19]. Interestingly, each set is connected to a particular internal state of the system and by changing the state, a different set of equations is triggered to describe dynamics of the system. This property is named "state-switching" and we suspect it beneficial in evolving solutions for some types of problems such as reactive tasks, in contrast with having a single set of differential equations as utilized in other GRN models.

2 Fractal Gene Regulatory Network

FGRN is inspired by biological cells [10]. In a biological cell, a number of genes (genome) encode proteins and the conditions of producing them. Proteins are means of interactions between the genes and also between the genome and the environment. These lifelong interactions drive the development and behaviors of the cell. The rate of production of a protein is based on the current protein content of the cell and environmental stimulants. For a particular gene, if a sufficient amount of specific set of proteins exists, the encoded protein is produced (or suppressed).

FGRN model is described in detail in [3, 18]. Here we give a very short summary stressing the points which are more important for this paper. An FGRN system contains "fractal proteins" and a set of genes which encode them. Fractal proteins are introduced as an abstraction of the interaction substance. A fractal protein consists of two parts: shape, and concentration level. The protein shape defines how the protein interacts in the system. The concentration level represents the current amount of the protein and is between zero and a maximum value. A level of zero means the protein doesn't exist at the moment.

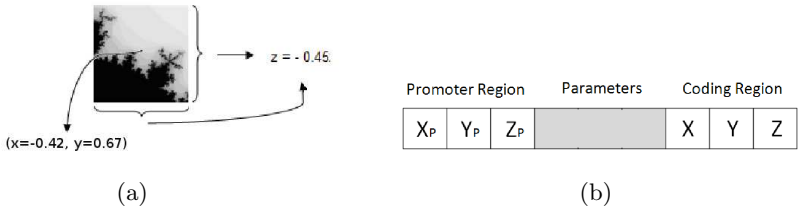


Fig. 1. An example fractal protein shape (a) and a typical gene (b)

A protein shape is a square window on the Mandelbrot fractal set and is encoded in a gene by three real values (x, y, z) (see Fig. 1(a)). By changing these values, the window reaches different locations and scales on the fractal set.

Genes in the FGRN system are of different types: environmental, receptor, regulatory, and behavioral. The genes belonging to the first three types encode the shape of proteins of the same type. The production rate of an environmental protein is determined by an environmental stimulant. This way, environmental information is brought into the system. Receptor proteins are responsible for filtering out parts of the environmental proteins. The production rate of regulatory proteins are regulated by a combination of proteins inside the system, namely a "protein compound". The behavioral genes generate outputs of the FGRN system.

All the environmental and regulatory proteins which are currently present inside the system merge together to a protein compound. The protein compound interacts with the regulatory and behavioral genes and regulates the concentration level of regulatory proteins and the output of behavioral genes.

A typical gene is a sequence of numbers. It consists of a promoter region, a coding region, and a set of parameters (Fig. 1(b)). The coding region encodes the shape of the protein producible by the gene. The promoter region encodes a protein shape as well. The present protein compound in the system matches against this shape and together with the parameter set of the gene, a value is computed. This value is used to determine the new concentration level of the gene in case of the regulatory genes and the output of the gene in case of the behavioral genes.

The lifetime of an FGRN system consists of several repetitions of the following steps: First, parts of the environmental proteins are filtered out by receptor proteins and the concentration levels of the rest are updated based on the environmental stimulants. Then the protein compound is computed from the present regulatory and environmental proteins. Then the new concentration levels of the regulatory proteins and the outputs of the system are produced based on the protein compound and the corresponding genes.

2.1 State-switching property

Here we aim to drive attention to an important characteristic of FGRN model we call "state-switching property". In order to do that, we leave the conventional

descriptive viewpoint of the model as an ongoing interactions between fractal proteins and genes. Instead, we view the system as several fixed conditional sets of differential equations (e.g. Table 1).

As mentioned before, a protein compound is a representation for a combination of the proteins currently present in the system. During lifetime of a system, the concentration levels of proteins may change. The proteins may vanish (their concentration reaches zero) or appear anew to the system (their concentration raises from zero). This means that the set of present proteins may change during the lifetime and consequently the protein compound can get different shapes.

We consider a set of present proteins as a particular state of the system. Since every single protein shape is encoded in the genome and doesn't change during lifetime, for every particular state of the system there is a fixed shape for the protein compound and it is computable solely based on the genome.

A new value for concentration level of a regulatory gene (or the output of a behavioral gene) is computed based on the promoter and parameters of that gene, and the present protein compound. A matching operation is performed between the shape of the compound and the protein shape encoded in the promoter region of the gene. From this matching operation, a contribution degree is computed and assigned to each protein which has participated in forming the protein compound. This contribution degree is a coefficient in a differential equation describing the new concentration level or output of the gene in the current state of the system. Therefore, for every particular state of the system a set of differential equations are defined. The variables of these equations are the concentration levels of the present proteins. In every step, the new outputs and regulatory concentration levels are computed based on the current set of the differential equations. If a change in the concentration levels leads to switching between the states of the system, a different set of differential equations is triggered in the next step. Note that these different conditional sets of equations are implicitly encoded in the FGRN genome and evolve during generations.

A detailed description about extracting the equational representation of the system from both the genome and definition of interactions in FGRN model is explained in [19]. An example description of a simple FGRN is represented in Table 1. The table describes a system with four different states (conditional statements) where a set of differential equations is associated with each state.

3 Investigated Scenario

In this work, populations of FGRN genomes are evolved as local controllers for a modular snake robot to perform a reactive locomotion task. The robot is made up of three homogenous modules and is supposed to locomote in a specific direction until it reaches a target zone at a random distance. The target zone is realized by a light source and the sensors of modules perceive the light only within an area with a threshold distance from the source. This area is considered the target zone and the controllers should react to that by preventing the robot from exiting the zone.

Table 1. An example of a simple FGRN as several conditional sets of differential equations. This system consists of four states. S represents current state as the set of present proteins. $P1$ and $P2$ correspond to an environmental and a regulatory protein respectively and $p1$ and $p2$ are their corresponding concentration levels. *out* is the output of the system.

condition (state)	equation set
if $S = \{P_1, P_2\}$	$p_2 \leftarrow 0.8p_2 - (0.2p_1 + 0.5p_2) \times tanh(0.6p_1 + 1.5p_2 - 3.6) - 0.2$ $out \leftarrow 0.15p_1 + 0.2p_2 + 4$
if $S = \{P_1\}$	$p_2 \leftarrow 0.4p_1 + 0.5$ $out \leftarrow 0.32p_1 + 2$
if $S = \{P_2\}$	$p_2 \leftarrow 0.8p_2 - 0.25p_2 \times tanh(0.5p_2 - 0.4) - 0.2$ $out \leftarrow 0$
if $S = \{\}$	$p_2 \leftarrow 0.4$ $out \leftarrow 0.25$

There is no explicit communication between the modules. The modules are controlled independently by genetically identical controllers. Since local sensors provide the input values for the modules, the outputs which are generated by the controllers may differ and lead to various behaviors for different modules.

The experiments are performed in Symbricator3D [16]. Symbricator3D is a simulator designed for the projects SYMBRION and REPLICATOR [15, 12] and uses the design of the prototype in [8] (Fig. 2). It is based on the game engine Delta-3D and uses the Open Dynamics Engine for the simulation of dynamics.

For the current experiments, three Symbricator3D modules are connected to each other to form a short snake. Each module is supplied by two proximity sensors at the front and rear faces which are implemented in the main simulator. In addition, we utilize an ideal luminance sensor concerning the target zone. The sensor provides a zero value when the light source is farther than a threshold. Otherwise the value is inversely proportional to the distance from the source.

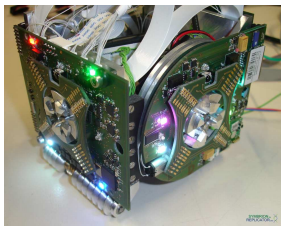


Fig. 2. A prototype of the module hardware

Since three sensors are used for a module, four types of environmental genes are defined, one for each sensor and another one provides a maternal protein which is always produced. A value received from a sensor is normalized and determines the concentration of the proteins encoded by the genes of the associated type. Every robotic module has an actuator which is a central hinge. All the behavioral genes are associated with this actuator. The total value produced

by the behavioral genes are considered to be the controller output. The output is scaled into the appropriate range and fed into the hinge making it to approach a specific angle.

Table 2. Evolutionary settings and the initial number of each type of genes

population size	#generations	#parents for crossover	mutation probability
30	25	12	1%
#receptor-genes	#evironmental genes	#regulatory genes	#behavioral genes
2	4	2	2

3.1 Evolving for the reactive behavior

A variant of a Genetic Algorithms is used in this experiment (See [3] for details). The initialization and evolutionary settings are represented in Table 2. A population of 30 randomly generated FGRNs is pre-evolved for locomotion in the right direction and this population is then evolved for the full behavior.

Fitness is computed based on two independent evaluations of a genome. For every evaluation, the target is set in a pre-specified distance from the robot. The two distances are different and fixed. The fitness is computed at the end of each evaluation period and the total fitness is the minimum of the two.

The reason for having two evaluations for each genome initiates from an evolutionary trap of the task. Having only one evaluation may end up to non-reactive controllers with high fitness evolved from exploiting the particular evaluation's settings, e.g. distance from the target zone and length of the evaluation period. For example, if a simple non-reactive controller makes locomotion with a particular speed, the robot happens to be at the target zone at the end of the evaluation period that leads to a high fitness.

Due to high computational costs in the simulator, the number of evaluations within reasonable time is limited. We set the number of evaluations of each genome to two which keeps the computational costs relatively low and still avoids the evolutionary trap. The fitness function of an evaluation is defined as follows:

$$fitness = \begin{cases} \alpha \times \frac{d_{TH}}{d_{current}} + g(speed_{out}, speed_{in}) & \text{if } d_{current} < d_{TH} \\ g(speed_{out}, speed_{in}) & \text{else} \end{cases} \quad (1)$$

where $d_{current}$ is the current distance from the center of the target zone, d_{TH} is the threshold distance of visibility, $speed_{out}$ and $speed_{in}$ are the speeds of locomotion outside and inside the target zone. α is a coefficient, and g is a function scoring for higher ratio of speeds outside to inside the zone.

3.2 Investigating the state-switching property

Two experiments with different FGRN setups are performed. In the first experiment, the standard FGRN setup is implemented. In the second setup, the

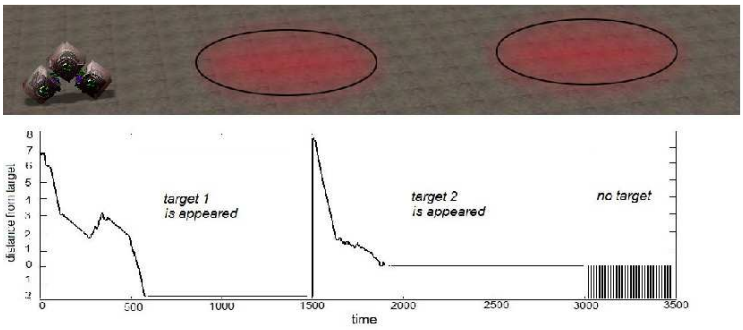


Fig. 3. Two consecutive random targets appear and disappear one after the other (top), and distance of the robot from the existing target during locomotion (bottom).

Table 3. Experimental settings and results

target distances during evolution	9	19	
target distances during reactivity evaluation	8	12	16	20	24 (units)
duration of evaluation period			
	during evolution	during reactivity evaluation	**successful runs**
state-switching on	600	3000	**80**%
state-switching off	1450	7000 (ticks)	**50**%
target visibility threshold 4 (units)			

state-switching property is suppressed by altering the model such that a single state and consequently a single set of equations are always triggered during runtime. In order to do that, all proteins of the system are counted as present proteins which determine the state of the system. Each experiment is repeated for 10 independent runs evolving for the full behavior in 25 generations. Every run starts from a population that is pre-evolved for locomotion.

Table 3 represents the settings including the duration of evaluation periods and the distances of target zones for both setups (state-switching on/off). Due to the various speeds of locomotion achieved by the pre-evolved populations of each setup, the durations of evaluation periods are different. The evolutionary progress of the evaluated fitness of the two setups are represented in Fig. 4 for the 10 independent evolutionary runs. As it is demonstrated in the figure, the setup with state-switching property (standard FGRN) performs better than the setup with the suppressed state-switching in terms of the evaluated fitness. Performing a Wilcoxon rank-sum test showed a difference between the two setups with a significance of 0.3 for both mean and best fitnesses in the last generation.

In order to evaluate the ability of the evolved controllers to adaptively react to their environment, the behavior of the robot achieved by the evolved controller in each run is observed against a set of additional differently-positioned targets. For each target, the robot starts from its initial position and the behavior is observed for a sufficiently long evaluation period. The distances of the targets and durations of evaluation periods are presented in Table 3.

The observations indicate that the robots of all the 10 runs of each setup react to the target zone by changing their movement when the zone is reached. For the setup with state-switching, in 8 runs out of the 10, the robot manages to successfully stay in the zone until the end of the evaluation period for all the five different targets. In some cases, the robot stands still at some point of the target zone, lying down on the ground or bending. In the other cases, the robot moves slowly back and forth in the way that it stays inside the zone. For the setup with suppressed state-switching, in 5 runs out of the 10, the robot manages to stay in the target zone for all the targets. Table 3 summarizes the settings and the results of the two investigated setups. As it is represented in the table, the standard FGRN setup shows a higher performance than the setup with the suppressed state-switching indicating the importance of the state-switching property in the FGRN system.

3.3 Observing a typical evolved controller

In order to look at the potential behaviors achieved as side-effects of evolution, a typical controller evolved in the standard FGRN setup is chosen from an arbitrary run. The behavior of the robot is observed in two observation settings.

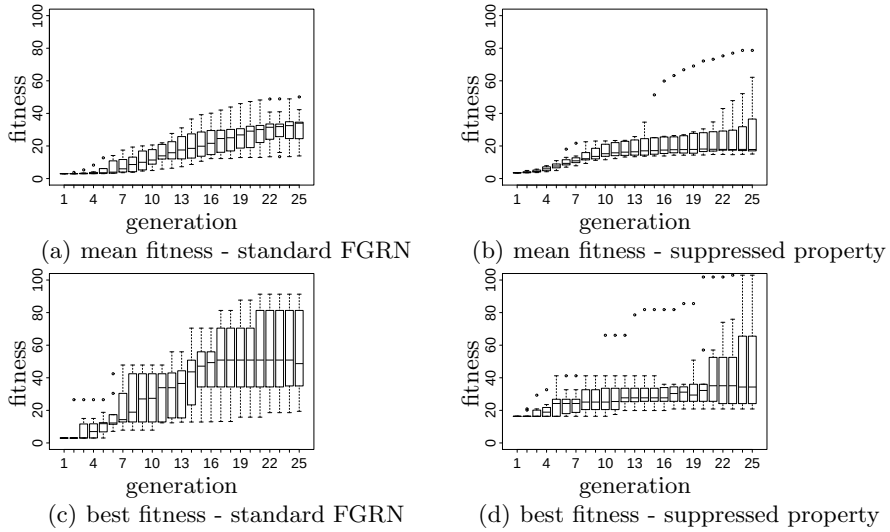
In the first setting, two consecutive targets are used (Fig. 3). The first target is positioned at a random distance from the robot and stays there for a long period of time (1500 sec). Then the target disappears and a new target appears farther at another random distance. For the observed controller, the robot reaches the first target zone and stays there as long as the target exists. Then, it continues the locomotion and reaches the second target and stays there. When the second target disappears as well (3000 sec), the robot starts to move again.

In the second observation setting, the robot is initially positioned in the target zone while the center of the zone is situated behind it. In this case, the robot moves backwards for a short distance. Then the target starts to move backwards in a slow speed. As a response to that, the robot also moves slowly backwards following the target[1]. Although these types of behaviors which are considered as side effects of an evolutionary run might be just random, they are achieved for free and still seem to be interesting.

4 Conclusion

This paper reported an application of FGRN system in controlling a modular snake-shaped robot in a reactive task. The task demands for appropriate changes in the robot behavior in response to particular environmental changes. In previous reactive controllers for multi-modular robots, e.g. HyperNEAT [6] and Central Pattern Generators (CPG) [11], or single robots ,e.g. Neural Networks (ANN) [20], the proper behavior is achieved by modulating an input part of an ANN to switch between the ordinary and special behaviors based on environmental signals. A relatively similar mechanism, namely “state-switching

[1] http://payam.zahadat.com/pub/reactiveController1.mpeg



(a) mean fitness - standard FGRN (b) mean fitness - suppressed property

(c) best fitness - standard FGRN (d) best fitness - suppressed property

Fig. 4. Progress of the fitness values during evolution.

property", is highlighted here as a characteristic exclusive for FGRN model in contrast with other GRN models [13, 1, 5, 4]. As demonstrated in the paper, this capability of implicitly switching between internal states is beneficial when FGRNs are evolved for reactive tasks. The FGRN systems were evolved for several independent evolutionary runs and post-evaluations of the achieved controllers demonstrated successful reactive behaviors for the majority of the runs. In addition, the behavior of an arbitrary evolved controller was observed in new environmental settings and interesting behaviors as side-effects of evolutionary process were detected. The evolvability of the model for adaptive reactions of higher complexity will be studied in future. In order to investigate the effect of "state-switching property" of FGRNs in evolving reactive controllers, an altered version of the model was implemented such that the state-switching property is suppressed. The altered FGRN was evolved for the same task and the comparison of the resulted controllers indicated the significance of "state-switching property" in achieving the successful solutions for the investigated task. In the next step of this study, inclusion of this characteristic in other GRN models will be investigated in order to achieve potential improvements in GRNs.

Acknowledgments. The authors thank Heiko Hamann and Jürgen Stradner for their helps and comments during this work. This work is supported by: EU-IST-FET project 'SYMBRION', no. 216342; EU-ICT project 'REPLICATOR', no. 216240; Austrian Federal Ministry of Science and Research (BM.W_F); EU-ICT 'CoCoRo', no. 270382.

References

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

2. Bentley, P.J.: Adaptive fractal gene regulatory networks for robot control. In: Workshop on Regeneration and Learning in Developmental Systems in the Genetic and Evolutionary Computation Conference (GECCO 2004) (2004)
3. Bentley, P.J.: Fractal proteins. J Genet. Program Evol. Mach (5), 71–101 (2004)
4. Bongard, J.C., Pfeifer, R.: Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. pp. 829–836. Morgan Kaufmann (2001)
5. Eggenberger, P.: Evolving morphologies of simulated 3d organisms based on differential gene expression. In: Proceedings of the fourth european conference on Artificial Life. pp. 205–213. MIT Press (1997)
6. Haasdijk, E., Rusu, A.A., Eiben, A.: Hyperneat for locomotion control in modular robots. In: 9th International Conference on Evolvable Systems (ICES 2010). pp. 169–180 (2010)
7. Hamann, H., Stradner, J., Schmickl, T., Crailsheim, K.: Artificial hormone reaction networks: Towards higher evolvability in evolutionary multi-modular robotics. In: Proc. of the ALife XII Conference. pp. 773–780 (2010)
8. Harada, K., PaoloCorradi, Popesku, S., Liedke, J.: Reconfigurable heterogeneous mechanical modules. In: Levi, P., Kernbach, S. (eds.) Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution. Springer-Verlag (2010)
9. Krohn, J., Gorse, D.: Fractal gene regulatory networks for control of nonlinear systems. In: Lecture Notes in Computer Science, vol. 6239, pp. 209–218. Springer-Verlag (2010)
10. Lodish, H., Berk, A., Zipursky, L.S., Matsudaira, P., Baltimore, D., Darnell, J.E.: Molecular Cell Biology. W.H. Freeman and Company, Madison Avenue, New York, USA, fifth edn. (2003)
11. Manoonpong, P., Pasemann, F., Roth, H.: Modular reactive neurocontrol for biologically-inspired walking machines. The International Journal of Robotics Research 26, 301–331 (2007)
12. REPLICATOR: Project website (2011), http://www.replicators.eu
13. Roggen, D., Federici, D.: Multi-cellular development: is there scalability and robustness to gain? In: Parallel Problem Solving from Nature 8 (PPSN'2004). pp. 391–400 (2004)
14. Schmickl, T., Hamann, H., Crailsheim, K.: Modelling a hormone-inspired controller for individual- and multi-modular robotic systems. Mathematical and Computer Modelling of Dynamical Systems 17(3), 221–242 (2011)
15. SYMBRION: Project website (2011), http://www.symbrion.eu
16. Winkler, L., Wörn, H.: Symbricator3D - A distributed simulation environment for modular robots. In: ICIRA. pp. 1266–1277 (2009)
17. Zahadat, P., Christensen, D., Schultz, U.P., Katebi, S., Støy, K.: Fractal gene regulatory networks for robust locomotion control of modular robots. In: From Animals to Animats 11. pp. 544–554 (2010)
18. Zahadat, P., Katebi, S.D.: Tartarus and fractal gene regulatory networks with inputs. Advances in Complex Systems (ACS) 11(06), 803–829 (2008)
19. Zahadat, P., Støy, K.: An alternative representation of fractal gene regulatory networks facilitating analysis and interpretation. Annals of Mathematics and Artificial Intelligence (Submitted)
20. Ziemke, T., Thieme, M.: Neuromodulation of Reactive Sensorimotor Mappings as a Short-Term Memory Mechanism in Delayed Response Tasks. Adaptive Behavior 10(3-4), 185–199 (2002)