Fractal gene regulatory networks for robust locomotion control of modular robots

Payam Zahadat[1 2], David Johan Christensen[1], Ulrik Pagh Schultz[1], Serajeddin Katebi[2], and Kasper Stoy[1]

[1]The Maersk Mc-Kinney Moller Institute,
University of Southern Denmark
{paza, david, ups, kaspers}@mmmi.sdu.dk
[2] Department of Computer Science and Engineering,
School of Engineering, Shiraz University, Shiraz, Iran
{zahadat, katebi}@shirazu.ac.ir

Abstract. Designing controllers for modular robots is difficult due to the distributed and dynamic nature of the robots. In this paper fractal gene regulatory networks are evolved to control modular robots in a distributed way. Experiments with different morphologies of modular robot are performed and the results show good performance compared to previous results achieved using learning methods. Furthermore, some experiments are performed to investigate evolvability of the achieved solutions in the case of module failure and it is shown that the system is capable of come up with new effective solutions.

Keywords: Fractal Gene Regulatory Networks, Modular Robots, Robot Control, Evolutionary Computation.

1 Introduction

The purpose of this paper is to investigate the capability of Fractal Gene Regulatory Networks (FGRNs) to control modular robots. FGRN [1] is a special type of computational Gene Regulatory Networks (GRNs) which utilizes fractal proteins to interact with a genotype. Modular robots are robots built from a number of mechanically coupled modules which can connect in different ways and each module is controlled by its own local controller. They have the potential to be versatile and robust, but due to their distributed, dynamic nature they are difficult to control.

Complex successful living phenotypes can be found everywhere in nature. Many of them consist of several cells each performing its own function related to position and role in the phenotype. Nature employs a complicated process of indirect mapping to develop a complete multi-cellular phenotype from a genotypic code. Instead of direct phenotype-genotype mapping normally used in conventional Evolutionary Computation (EC), the lifelong process of natural development is controlled by an ongoing interaction between genotype and intermediate substrates called proteins which are encoded by the genotype. This interaction is considered a network of genes which is called Gene Regulatory Network.

AN FGRN cell contains a genotype –called genome- that encodes fractal proteins, and a compound substrate -called cytoplasm- that maintains the proteins inside the cell. Developmental process of a cell is controlled by interaction between cytoplasm and genome. The process can be affected by information provided by the environment which is also represented by fractal proteins. During the lifetime of the FGRN cell, complex output patterns can be produced and used for different purposes such as controlling robots [2, 3, 4].

FGRN systems can be implemented distributedly. In a distributed system, all cells use the same genotype, but they run in parallel to each other. By providing proper environmental information for each cell, different cells in a system might follow different developmental trends and make appropriate output patterns. This distributed nature of the system potentially makes it suitable for controlling modular robots since each FGRN cell can be used to control one module of the robot. But the question is if this works in practice and what properties the resulting system has.

Modular robots are resource-constrained. They usually have little processing power and low inter-module communication abilities. In addition, based on the dynamic nature of a modular robot, failures might happen in modules, they can break, or the user may take apart the robot or detach some modules for different reasons while the robot is still supposed to work. In designing controllers for modular robots, it is desirable to have an acceptable level of robustness encountering these properties along with the characteristics of scalability, usability in different morphologies, and biological plausibility.

Distributed control of ATRON modular robots [5] which are supposed to perform a locomotion task is investigated here. Three different morphological configurations of the robot are used as experimental case studies. Previous works [6] have shown good performances for learning methods in these cases. The results achieved here, demonstrates evolvability of FGRN systems as distributed controllers of the robots which is the first step towards implementing FGRN systems to cope with more complex challenges in modular robots.

In an additional experiment, one of the three robots is selected and the behavior of the evolutionary system after a module failure is investigated. The results demonstrate that the FGRN system is evolvable to find new solutions for the new situations.

2 Related Works

In the field of computation systems different approaches have been used to create evolutionary systems with a developmental process for genotype to phenotype mapping. In some works, models of GRNs are evolved for making mathematical output functions such as sinusoid, exponential and sigmoid [7]. Some researchers have designed GRN systems for developing neural networks for controlling robots [8, 9, 10] or specifying the morphology of 3D organisms [11]. Also, GRN models have been used to develop the morphology of robots as well as their neural network controllers [12]. Other models of GRNs have been proposed in [13, 14, 15, 16].

In a model of GRN called FGRN, Bentley [1] introduces fractal proteins as an intermediate substrate that resembles the role of proteins in the cell. The recursive and

self-similar nature of fractal proteins make the fractal genetic space evolvable, complex, and redundant [2, 3, 17]. FGRNs are evolved to produce desired patterns [18], controlling conventional robot and motion planning [2, 4]. On the other hand, in the field of robotics, different approaches have been investigated by researchers to control modular robots. Co-evolving morphology and control of simulated modular robots [19, 20], learning strategies [6, 21] and applying central pattern generators to control modular robots [22, 23] are some of the reported researches in the field.

3 Gene Regulatory Networks

3.1 Biological Inspiration

Development of phenotypes can be thought of as a product of interaction between genes and proteins in their environment. Almost everything inside a cell is carried out by proteins. Proteins drive development and functioning of a cell and are used for communication between cell and its environment that might include other cells.

A cell contains a genome and a cytoplasm which are surrounded by a membrane (Fig. 1). The membrane separates the interior of a cell from the outside environment. Receptor proteins are embedded in the membrane and control the movement of environmental proteins into the cell. The cytoplasm contains a compound of proteins inside the cell. The genome consists of a set of genes. Every gene contains a sequence that encodes a protein (coding region) and a sequence that determines the conditions for activation or suppression of that gene (promoter region) (Fig. 1).

Fig. 1. An example cell (left) and a gene (right).

An active gene expresses and produces its appropriate protein as encoded in its coding region. For a gene to be activated, a proper amount of appropriate protein compounds in cytoplasm must match the promoter region of the gene.

The cytoplasm content is altered by proteins produced by genes inside the cell or the environmental proteins which have entered the cell passing through receptors.

During the development of a cell, the protein content of the cytoplasm might match against the promoter of some genes and get them to suppress or express proteins.

Every produced protein will merge to the cytoplasm and would alter its content. The new content, in turn, affects the expression of genes in the next step. It might cause new proteins to be produced or the amount of the current proteins in the cytoplasm to be changed. In this way, every gene which makes protein inside a cell

might influence the expression of other genes (including itself) directly or indirectly. In the same way, the proteins which enter the cell from the environment can influence expression of genes and participate in development of the cell. On the other hand, the functional behavior of a cell is determined by special proteins in the cell. These proteins may change the shape, structure, or other properties of the cell, or might be used as signals to the outside environment. Production of these proteins is determined by the corresponding genes and the protein content of the cytoplasm. Therefore, variations in cytoplasm content might lead to variations in the behavior of the cell to the outside world.

The ongoing interaction between proteins and genes continues for whole lifetime of a cell and is considered a network of genes which regulate the expression of each other and is called a Gene Regulatory Network (GRN).

3.2 Implementation

In a series of works reported by Bentley [1, 2, 3, 18] a protein model called fractal protein is developed as the protein substance of gene regulatory networks in an evolutionary system.

Each fractal protein is a square window on the Mandelbrot fractal set with a pre-specified resolution (Fig. 2). Fractal proteins are represented by a square matrix of integer values but can be encoded by only three values (x, y, z). (x, y) determine the center of the window on the fractal set. z specifies the length of the sides and can be inversely considered as the amount of magnification in the fractal image. Iterating Mandelbrot formula along with the three values specifies every entry in the matrix of a fractal protein and determines the image. Fig. 2 shows an example fractal protein.

In addition to a square matrix of integer values, a single integer value relates to each fractal protein as its concentration level. The concentration level represents the current amount of the protein. The value increases when more of the protein is produced and decreases slowly over time to resemble normal degradation that happens in real cells.

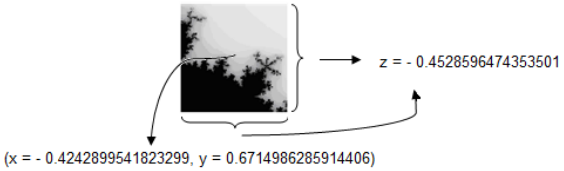

Fig. 2. An example fractal protein and the three values which specify it.

Fractal proteins can merge together and make protein compounds. A fractal protein compound is represented by a square matrix of integer values in the same way as fractal proteins. In order to merge a protein into a protein compound, for every entry in the corresponding matrices, the winner is the paler pixel in the fractal image. See Fig. 3 for an example.

The cytoplasm of an FGRN cell is a compound of all the proteins inside the cell. Every protein that is produced in the cell or enters the cell from outside will be merged into the content of the cytoplasm.

Fig. 3. Two proteins (left and middle) are merged (right).

Fig. 4. The cytoplasm protein compound (left) matches against the promoter of a gene (middle) and the absolute difference is calculated from the result (right).

Fig. 5. Environmental protein (left) passes through the receptor protein (middle) and some portion of it (right) is allowed to enter the cytoplasm.

A genome in an FGRN cell consists of a set of genes. Genes consist of a sequence of values representing promoter region, coding region, threshold parameters, and type of the gene.

The coding region contains the three real values which encode a fractal protein. In the same way as the coding region, the promoter region consists of three real values that encode a square matrix of fractal values as well. This matrix works as a window that will be put on the cytoplasm protein compound matrix and is used to calculate the matching degree between the promoter of the gene and cytoplasm content (See Fig. 4 for an example). The matching degree along with the total concentration of matched proteins on promoter region, determine the degree of activation (or suppression) of the gene and might specify its protein production rate. Threshold parameters are used to calculate the matching degree and protein production rate of each gene.

To assimilate different types of genes in a cell, each gene contains an integer value representing its type. Every gene belongs to one of the following types:

- Regulatory gene, which comprises both promoter and coding region. Its encoded protein will be produced and merged into cytoplasm and participate in regulation of expression (or repression) of genes.

- Environmental gene, determines the proteins which might be present in the environment of the cell.
- Cell receptor gene, contains a coding region and produces a receptor protein. Receptor proteins merge together and act as a mask to permit variable portions of environmental proteins to the cytoplasm (See Fig. 5).
- Behavioral gene, which comprises a promoter region and a coding region. The values in the coding region can directly participate to determine the outputs of the cell.

Lifetime of an FGRN cell consists of a number of developmental cycles which can be summarized as the following steps:

- Produce receptor and environmental proteins.
- Pass the environmental proteins through receptors and merge them into the cytoplasm content.
- For every behavioral and regulatory gene,
  - If the content of cytoplasm matches the promoter,
    - If the gene is behavioral: utilize the coding region to specify the cell's outputs
    - If the gene is regulatory: express the coding region and merge the produced protein into the cytoplasm
- Update concentration level of proteins in the cytoplasm.

For more detailed descriptions of FGRN systems and the corresponding formulas see [1, 2, 4].

4 Evolving FGRNs to control modular robots

Every module of a robot is considered a cell in a multi-cellular creature. Each module contains an FGRN cell which includes its genome and cytoplasm. All the FGRN cells run in parallel and independent of each other and make their own sequence of output commands for the modules containing them.

All the cells are genetically identical which means they contain an identical copy of a genome. Environmental information about the number of connections and the initial orientation of the module which contains the cell is provided for each cell in the form of environmental proteins. Therefore, two cells which are contained in two modules with different environmental situations initially contain different proteins in their cytoplasm. Different cytoplasm content might activate different genes of the genome of each cell and leads to different internal interactions and developmental trends. Consequently, while the cells are genetically identical, different phenotypic characteristics might be formed and different output commands might be generated by the cells during their lifelong development.

In this work proper genomes are evolved such that when they are copied in all the cells of one modular robot, each cell can generate a right sequence of commands for its module using the appropriate environmental information and make the robot perform its locomotion task.

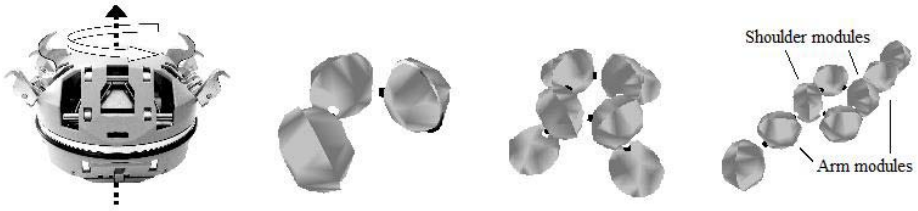

Fig. 6. From left to right: An ATRON module, Two-wheeler, Quadrupedal, and Crawler robots.

4.1 Experimental Setup

Robot Simulator Simulation experiments are performed in an open-source simulator named Unified Simulator for Self-Reconfigurable Robots (USSR) [24]. The simulator provides physics-based simulations of different modular robots including the ATRON robot. The ATRON robot is a homogenous, lattice-based self-reconfigurable modular robot. An ATRON module weighs 0.850kg and has a diameter of 110mm. A module consists of two hemispheres which can rotate infinite relative to each other with a speed of 360 degrees in six seconds. Each hemisphere contains two passive (bars) and two active connectors (hooks), see Fig. 6.

Table 1. Genetic parameters.

#population size	#generations	crossover rate	mutation rate
20	50	40 %	1 %

# regulatory genes	#receptor genes	#environmental genes	# behavioral genes
4	5	9 / 10 (for crawler)	1

Genetic and developmental configurations A population of 20 FGRN genomes is evolved for 50 generations using a variant of steady-state genetic algorithm with lifespan limits [1]. Each genome is initialized with randomly generated regulatory, receptor, environmental, and behavioral genes. The initial number of each type of gene and the genetic parameters are shown in Table 1. Evolution is allowed to regulate the number of each type of genes (See [1, 4] for more details).

To evaluate a genome, identical versions of a genome are copied to all the modules' FGRN cells. Each cell receives some environmental proteins describing the number of connections and the initial orientation of the module in which it is situated. Also an additional environmental protein common between all the cells is initially provided.

In order to make an action for each module in every step, modules independently run their own FGRN cell for one developmental cycle and receive an output from the cell. The cell output is calculated on the basis of activation of behavioral genes inside

the cell. The output is mapped to one of the following three commands that will be performed by the module in that step:

- rotateRight – rotates clockwise 90 degrees
- rotateLeft – rotates counterclockwise 90 degrees
- stop – rotate zero degrees

After a specified time span (50 sec.), fitness is evaluated as the distance between the initial position and the end position of center of mass of the robot.

The run-time procedure of a robot can be summarized as follows:

- Create genome
- For every Module of the robot:
  - Make an empty FGRN cell and put a copy of the genome into it.
  - During the run-time of the robot:
    - Receive information about the module's environment and activate the relevant environmental proteins.
    - Develop the cell for one cycle according to the developmental steps in section 3.2 and receive cell output.
    - Translate cell output to the module command and
    - Execute the command.

Case studies We have evolved multi-cellular FGRN controllers for three robots with different morphologies and the same genetic configurations. Fig. 6 shows the three morphologies which are used. In order to keep things as simple as possible we didn't use any communication between modules. As it might be expected, for the two-wheeler robot, evolution leads to controllers which rotate the two opposite modules in the opposite directions to move the robot like a car. For the quadrupedal robot, a swimming-like behavior evolved. For the crawler robot, different crawling gaits evolved. In order to evaluate the robots, the velocity of the locomotion is calculated for each robot. The best and population-average velocities are shown in Fig. 7. The figure shows the results averaged over 10 independent runs. The results are compared with the results achieved by a learning strategy reported in [6]. Table 2 shows the higher velocities achieved by the FGRN controllers and the learning controllers. The learning strategy is reinforcement learning accelerated by a heuristic which detects and repeats potentially underestimated actions to accelerate the estimation accuracy and presumably accelerates the learning.

In another experiment, the evolvability of the FGRN system is investigated after a module failure. The crawler robot is selected for this experiment. We considered the solutions found in the last experiment. Different gaits were recognizable between the solutions evolved in the 10 runs. Based on the position of the modules which had more effect in the locomotion, the solutions can be categorized in two main groups-solutions which mainly use the shoulder modules and solutions which mainly use the arm modules (See Fig. 6). The second group which has the velocity of higher than average-velocity is selected. In order to resemble a situation of failure, one of the modules of high importance (one of the arms) is disabled while the robot uses the previously evolved FGRN controller. Since the controller is not suitable for this new situation, the fitness falls considerably. Afterwards, the controllers are allowed to

evolve for 30 generations and the velocities of the new solutions are evaluated. As it is shown in Fig. 7, the velocity of robots falls after failure, and then rises when evolution continues. The performance of the new evolved controllers is investigated for robots both with the broken module and intact module (after repairing the broken module). Table 3 shows the velocities in different situations and represents a good performance for the new evolved controllers in both cases of intact and broken modules. Furthermore, the experiment repeated with the broken module to evolve controllers from scratch (See Fig. 7). The velocities are averaged over 10 runs of evolution (Table 3).

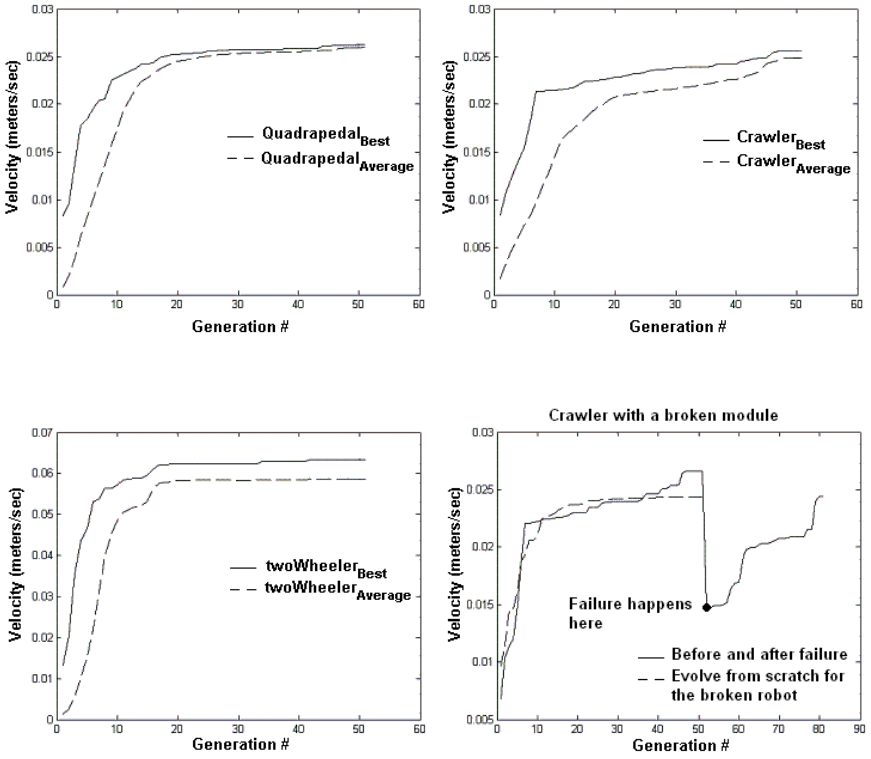

Fig. 7. Robot velocities for the three robots, and velocities of the crawler with a broken module.

Table 2. Comparison of the best velocities achieved by FGRN and [6] learning algorithm.

Robot Configuration	Learning [6]	FGRN (Population average)	
	Mean	Mean	Standard deviation
Quadrupedal	0.0208	0.0260	0.0011
Crawler	0.0210	0.0248	0.0038
Two-wheeler	0.0383	0.0586	0.0007

Table 3. Averaged velocities of the Crawler robot.

Velocity (all runs)	Before failure (selected runs)	After failure (selected runs)
0.0248	0.0262	0.0145

After more evolution (selected runs) – robot with failed module	After more evolution (selected runs) – module repaired	Velocity for broken robots evolved from scratch (all runs)
0. 0245	0.0235	0.0244

5 Conclusion

In this paper, we explored application of FGRN systems to control of modular robots. FGRN systems are inspired by natural cells and due to their internal interactions are able to generate complex output patterns which might be used as control commands. Implementing the FGRNs in multi-cellular way provides us a distributed controller for ATRON modular robots. The local controllers for all modules are encoded identically and run independently. In order to keep the system as simple as possible, there is no communication between modules in the current implementation. Communication between modules and different sensory information might be included in the future works.

We carried out experiments with different morphologies of the ATRON in a locomotion task and reached good performances. Results are compared to the previously reported results of robots employing a reinforcement learning strategy. Furthermore, we investigated the capability of the FGRN system to evolve more in case of a failure and the achieved controllers are evaluated for both intact and broken robots. The results show that the FGRN system is still evolvable to find new solutions for new situations of the robot.

Acknowledgments. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 - Future Emerging Technologies, Embodied Intelligence, under grant agreement no. 231688.

References

1. Bentley, P.J.: Fractal proteins. J. Genetic Programming and Evolvable Machines, vol.5 no.1, pp. 71--101. Springer (2004)
2. Bentley, P.J.: Adaptive Fractal Gene Regulatory Networks for Robot Control. In: Genetic and Evolutionary Computation Conference, Seattle, USA (2004)
3. Bentley, P.J.: Evolving Fractal Gene Regulatory Networks for Graceful Degradation of Software. In: Self-star Properties in Complex Information Systems, LNCS, Springer (2005)
4. Zahadat, P., Katebi S.D.: Tartarus and Fractal Gene Regulatory Networks with Input. J. Adv. Complex Sys, vol. 11, no. 6, pp. 803--829, World Scientific (2008)
5. Ostergaard, E.H., Kassow, K., Beck, R., Lund, H.H.: Design of the ATRON Lattice-Based Self-Reconfigurable Robot, J. Auton. Robots, 21(2), pp.165--183, (2006)

6. Christensen, D.J., Bordignon, M., Schultz, U.P., Shaikh, D., Stoy, K.: Morphology Independent Learning in Modular Robots, In: International Symposium on Distributed Autonomous Robotic Systems, pp. 379--391, (2008)
7. Kuo, P.D., Leier, A., Banzhaf, W.: Evolving Dynamics in an Artificial Regulatory Network Model, LNCS, vol. 3242, pp. 571--580, Springer-Verlag (2004)
8. Jakobi, N.: Harnessing Morphogenesis, In: International Conference on Information Processing in Cells and Tissues, Paton, R. Ed., Liverpool, UK., pp. 29--41, (1995).
9. Dellaert, F., Beer, R.: A Developmental Model for the Evolution of Complete Autonomous Agents, In: fourth international Conference on Simulation of Adaptive Behavior, Cambridge, MA, pp. 393--401, MIT Press (1996).
10. Federici, D.: Evolving a Neurocontroller through a Process of Embryogeny, In: Eighth International Conference of Simulation and Adaptive Behavior, Schaal, S. et al. (eds.), pp 373--384, MIT Press (2004).
11. Eggenberger, P.: Evolving Morphologies of Simulated 3D Organisms Based on Differential Gene Expression, In: Proc. 4th European Conference on Artificial Life (ECAL). Husbands, P., Harvey, I., (eds.), pp. 205-213, MIT Press, Cambridge (1997)
12. Bongard, J. C., Pfeifer R.: Evolving Complete Agents Using Artificial Ontogeny. In: Hara, F., Pfeifer, R. (eds.), Morpho-functional Machines: The New Species (Designing Embodied Intelligence), pp. 237—258, Springer-Verlag (2003)
13. Federici, D., Downing, K.: Evolution and Development of a Multi-Cellular Organism: Scalability, Resilience and Neutral Complexification, J. Artificial Life, vol. 12 (3), pp. 381-409, (2006).
14. Banzhaf, W., On evolutionary design, embodiment and artificial regulatory networks, in Embodied Artificial Intelligence, F. Iida, R. Pfeifer, L. Steels, and Y. Kuniyoshi, Eds., pp. 284--292, Springer (2004)
15. Hornby, G.S., Pollak, B.: The Advantages of Generative Grammatical Encodings for Physical Design. In Congress on Evolutionary Computation, IEEE Press, pp 600—607 (2001).
16. Kennedy, P.J., Osborn, T.R.: A Model of Gene Expression and Regulation in an Artificial Cellular Organism, J. Complex Systems vol. 13, no. 1, pp. 1--28, (2001).
17. Bentley, P.J.: Methods for Improving Simulations of Biological Systems: Systemic Computation and Fractal Proteins. J. R Soc Interface (2009)
18. Bentley, P.J.: Evolving fractal proteins, In 5th International Conference on Evolvable Systems: from Biology to Hardware, vol. 2606, pp. 81--92, Springer (2003)
19. Sims, K.: Evolving 3d morphology and behavior by competition. In: Proc. Artificial Life IV, R. Brooks and P. Maes, (eds.), pp. 28--39, MIT Press (1994).
20. Marbach D., Ijspeert, A.J.: Co-evolution of Configuration and Control for Homogenous Modular Robots, In: Proc. 8th International Conference on Intelligent Autonomous Systems, pp. 712--719, Amsterdam, Holland (2004)
21. Maes, P., Brooks, R. A.: Learning to Coordinate Behaviors, In: National Conference on Artificial Intelligence, pp. 796--802 (1990)
22. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic Locomotion Design and Experiments for a Modular Robotic System, J. IEEE/ASME Transactions on Mechatronics, vol. 10, no. 3, pp. 314--325, (2005)
23. Sproewitz, A. , Moeckel, R., Maye, J., Ijspeert, A.: Learning to Move in Modular Robots using Central Pattern Generators and Online Optimization, J. Rob. Res., vol. 27, no.3-4, pp. 423--443, (2008)
24. Christensen, D.J., Schultz, U.P., Brandt, D., Stoy, K.: A Unified Simulator for Self-reconfigurable Robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, (2008)