Co-evolution of Configuration and Control for Homogenous Modular Robots

Daniel MARBACH and Auke Jan IJSPEERT
Swiss Federal Institute of Technology at Lausanne,
CH 1015 Lausanne, Switzerland
Daniel.Marbach@epfl.ch, Auke.Ijspeert@epfl.ch
WWW: http://birg.epfl.ch

Abstract. Modular robots are well suited to implement key features of autonomous machines such as versatility, adaptability and reliability. Our vision is to tackle this task following the three major axes (phylogeny, ontogeny and epigenesis) that underlie the emergence of autonomous and self-organizing organisms in nature.

This paper presents Adam, our modular robot simulation and evolution tool. Adam successfully implements the first step of our project, inspired by the phylogenetic axis. We co-evolve configuration and control of locomoting homogenous modular robots by means of genetic programming. A tree-based genotype is used, encoding the control as well as the configuration of the modules. The modular robots are evaluated in a simulator that accurately models rigid body dynamics. Furthermore, we propose a grammar for an intuitive script that allows building modular robots 'by hand' or inspecting and manipulating evolved individuals.

1. Introduction

This paper presents Adam, a modular robot simulation and evolution tool. Our goal is a realistic simulation of autonomous modular robots by implementing the three axes of the POE model: Phylogeny (evolution), ontogeny (development) and epigenesis (learning). The project has only recently started consequently, we can only present preliminary results that touch exclusively the phylogenetic axis. We would like to stress that Adam is a modular robotics and not an artificial life project. Even though we are not currently working on a hardware prototype, we want our results to be theoretically transferable to reality. Therefore we aim at a realistic simulation using modules with flat connection surfaces similar to existing hardware and we evolve the configuration of the robot using this predefined module type.

1.1 Modular Robotics

Modular robots offer many interesting qualities of autonomous systems such as versatility, adaptability and reliability. With the number of modules used, the number of possible configurations grows exponentially and the number of degrees of freedom linearly, making modular robots extremely versatile. Modular self-reconfiguring robots can even adapt autonomously to new environments and tasks.

Reliability stems from redundancy. Modular robots that are controlled in a distributed manner are robust because failure of some modules only degrades the overall function. Furthermore self-repair mechanisms can be implemented by ejecting damaged modules and replacing them with spare ones or by self-reconfiguring the robot around them.

Another advantage of modular robots that is often pointed out in the literature is low cost. Mass production could lower the price of single modules but they will also be needed in great numbers to form a single robot. On the other hand, versatility and reliability are qualities of great commercial interest because the same robotic system could be used for diverse and changing tasks with low maintenance costs.

1.2 The POE Model

The POE model, introduced by Sipper et al. [1, 2], is an attempt to classify bio-inspired methodologies in design of computing machines. Considering life on earth, there are essentially three biological models of self-organization that explain the emergence of complex, autonomous organisms with such desirable qualities as growth, adaptability, fault tolerance, regeneration and reproduction: Phylogeny (P), Ontogeny (O) and Epigenesis (E).

Bio-inspired engineering methods can be classified along these three axes. We agree with Sipper et al. that novel bio-inspired systems can be obtained by combining two or ideally all three POE axes. An example of such a system would be an evolving (phylogeny) and self-replicating (ontogeny) modular robot with a neural network controller implementing reinforcement learning (epigenesis).

2. Related Work

We argue that modular robotics (also sometimes called cellular robotics [26]) is a perfect framework to build one day a truly autonomous machine by applying bio-inspired methodologies. As mentioned before, the Adam project has only just started. Our work so far concerns co-evolution of configuration and control and is situated exclusively on the phylogenetic axis. In this chapter, we focus on this field but we also present other related work that lets us hope, that it is indeed possible to build one day a POEtic modular robot.

Numerous research groups are working on modular robot hardware systems. Chain robots, for example M-TRAN [3], Polybot [4] or CONRO [5], are formed from chains of modules and have demonstrated re-configuration for various types of locomotion. On the other hand, lattice modular robotic systems like Telecube [6], Crystalline [7], I-Cubes [8], Fractum [9] or ATRON [10] locomote and reconfigure by moving modules to neighboring positions on a lattice. Swarmbots [11] uses small robots that can move autonomously and dock to form larger structures. Intelligence should emerge through interaction, like in swarms of social insects.

The complexity of designing the structure and programming the controller of the robot grows exponentially with the number of modules. Co-evolutionary algorithms have proved to successfully evolve morphology and controllers simultaneously to suit a particular task. Sims co-evolved body and brain for locomoting and competing block creatures already in 1994 [12]. The genotype, describing morphology as well as the neural system that controls the movements, is structured as a directed graph of nodes and connections, allowing repeating or recursive components. Ventrella evolved walking stick creatures [13]. Framsticks, a three-dimensional life simulation project, offers various genotypes and fitness functions to co-evolve morphology and control of virtual stick creatures [14]. Hornby and Pollack compared direct and generative representations for body-brain co-evolution and found that the latter ones achieve better performance [15]. Using L-systems as generative encoding, they evolved neural controlled robots that are more complex and use more parts than those in previous work. Bongard and Pfeifers creatures can be situated on the PO plane [16]. They evolve gene expression rules and simulate an ontogenetic process based on suc-

cessive divisions of body segments, involving both, the morphology and the neural controller.

The research cited above has in common, that the evolved robots or creatures are purely virtual. The mainly stick-like body-segments are themselves subject to evolution and do not correspond to actual hardware modules. In GOLEM, evolved stick creatures are brought to life with 3D printing technology [17]. The disadvantage of this approach is that the body segments need to be produced especially for every robot.

The philosophy of Adam is to use only one predefined module type, ideally modeling existing hardware. While the projects mentioned above also evolve the morphology of the body segments, we only evolve the configuration of the homogenous modular robot. With regard to the evolutionary algorithm this means a trade-off between a smaller phenotype space and more speed. Similar work has been done with LEGO robots [e.g. 18], but since many different module types were used, these robots are not well suited to implement key features of modular robots like self-reconfiguration or self-repair.

A distributed controller programming [19] and an evolutionary motion synthesis [20] method were proposed for the modular self-reconfigurable robot M-TRAN [3]. The simulation and the type of modules used are very similar to our work but while the first method doesn't apply evolutionary algorithms, the latter one only evolves the control for specific initial configurations that are not subject to evolution.

Symbiotic co-evolution between populations of initial configurations and controllers was realized with a simulation of ATRON [10]. However, due to the high complexity of the controllers of this lattice modular robot, only reaching and not locomoting behavior could be observed.

We conclude that, to our knowledge, Adam is the first project that co-evolves configuration and control of homogenous chain type modular robots, using modules with flat connection surfaces rather than stick-like body segments.

Interesting research situated on the ontogenetic and epigenetic axis encourages us for our future work. Projects involving self-replicating, self-assembling and self-repairing modular robots can be found in [21, 22, 23]. An example of the combination of evolution and learning to achieve complex and adaptive behaviors is [24].

3. Adam

As we mentioned in the previous chapter, the main difference between Adam and other projects that co-evolve modular robots is that we only evolve the configuration and not the morphology of the modules. Even though we do not have currently a hardware prototype, Adam robots should theoretically be buildable with corresponding modules. Adam uses hinge modules, consisting of two cubes, fixed together with a hinge joint. Other elements can be attached at each one of the ten free faces. The hinge can be rigid, elastic or powered by a motor.

Similar module types have successfully been built and tested by leading modular robotics research projects [3, 4, 5]. These modules, unlike the stick-like body segments used in co-evolutionary simulations cited in the previous chapter, contain a joint, other modules being attached at flat connection surfaces. On the other hand, the stick-like modules used in previous research are usually rigid, joints being formed through connections with other segments. This approach does not correspond with the hardware built so far in modular robotics.

The simulation environment is implemented with Russell Smiths Open Dynamics Engine (ODE [29]). It accurately models rigid body dynamics (kinematics, gravity, friction, collisions etc) in a world that consists for the moment of an infinite plane.

3.1 Control

Currently, Adam robots are controlled in a simple manner, with harmonic oscillators for generating trajectories, and PD controllers for producing the torques to follow these trajectories. For a given element *i*, the desired angle θ_i trajectory is defined by the amplitude A_i, frequency f_i and phase ϕ_i of a sinus oscillation (Eq 1). The PD controller applies a torque T_i to get a rotation of the hinge that depends on the desired angle θ_i, the actual angle θ_a and the angular rate $\dot{\theta}_a$ (the derivative of the actual angle, Eq 2). α and β are positive constants that correspond to the gains of the PD controller.

$$\theta_i = A_i \sin(2\pi f_i t + \phi_i) \quad (1)$$

$$T_i = \alpha(\theta_i - \theta_a) - \beta\dot{\theta}_a \quad (2)$$

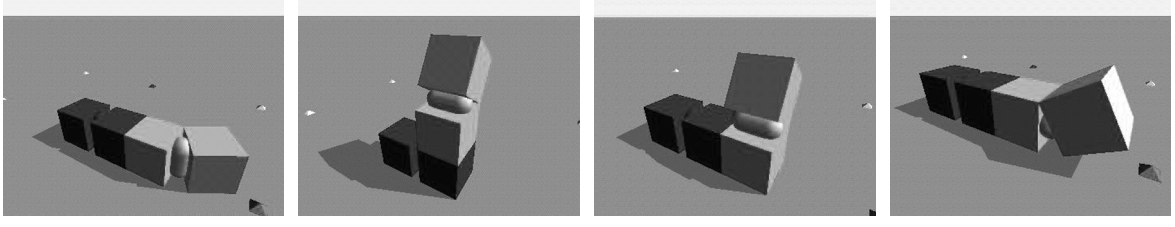
In this article, controllers will be defined by using a genetic algorithm to set the parameters A_i, f_i and ϕ_i for each element in a robot. This approach allows us to quickly explore different locomotion strategies in a relatively small search space. In the future, more complex nonlinear oscillators with coupling terms from proprioceptive sensors (e.g. joint-angle sensors, and torque sensors) will be used in order to develop controllers that can better deal with external perturbations.

3.2 The Adam Script

The user can define Adam robots with a simple script. Furthermore, this allows evolved robots to be saved in plain text format. They can then be inspected and edited by the user with the text editor of his choice.

The hinge module has ten faces where it is possible to attach other segments. These positions are defined within the local coordinate system of the hinge and are labeled as *P0* to *P9*. To attach a new module *Hb* to a hinge *Ha* that is already part of the unfinished structure, three parameters need to be specified. 1) The face of *Ha* where *Hb* should be attached; 2) The face of *Hb* that should be used; 3) The orientation that *Hb* must be fixed with, because there are four different ways to fix the two hinges with two specific faces. Examples of some elementary script expressions are illustrated in figure 1.

The Adam script is similar to Framsticks *recur* genotype [6] using bracketing to interpret the string as a tree, but it is much easier to read because it's designed for the user, and not for the genetic algorithm. The main differences are, that we separate the structure part (defining the configuration) from the one defining the control and that it is possible to declare body parts that can then be attached several times at various positions. Refer to figure 2 for an example of a complete script. A more detailed description of the whole script can be found on the Adam web page at: *http://birg.epfl.ch/page32031.html*



Ha Hb *Ha P5 (Hb)* *Ha P1 Hb* *Ha E Hb*

Figure 1: Some elementary script expressions. Hb has been assigned an initial angle of 30 degrees.

3.3 Co-evolution of Configuration and Control

The genetic encoding structures the phenotype space by defining which phenotypes are close genetically, i.e. separated by only few mutations. The fitness function induces another topology with respect to fitness values. Good encodings have a higher correlation between these two topologies [14, 25]. The Adam genotype is a tree, each node representing a module. Refer to figure 2. Obviously, such direct encodings strongly correlate the previously mentioned topologies because the genotype and the phenotype are closely related (actually, the robots genotype is equal with its internal representation in the simulation and evolution environment). Unfortunately, modular robots that contain cycles cannot be represented currently. Our tree-based genotype is very close to Sims generative encoding [12], with the difference that it is not yet possible to reuse components recursively in Adam.

Genetic operators include mutation and crossover. The mutation operator acts on parameters as well as on the structure of the robot. If mutation occurs on a numerical parameter, a random value from a normal distribution is added. The position and orientation that a module is fixed with are also subject to mutation. Furthermore, there's the chance of deleting limbs or attaching new, randomly initialized modules to free faces of the robot. Thanks to the tree structure of the genotype, the implementation of crossover is straightforward. A child is formed by copying the mother and replacing one of its sub trees with a sub tree of the father's. Obviously, crossover and mutation can generate invalid robots with intersecting modules. If this happens, the concerned robot is deleted and replaced with a new, randomly initialized individual. This is done in the hope of increasing diversity in the gene pool. At the beginning of the GA the population is initialized with randomly created robots. For selection and replacement we propose a rank-proportional roulette wheel method. When choosing a parent to produce offspring, the probability $p_s(i)$ for an individual i to be selected is inverse proportional to its rank $r(i)$ (the best robot has rank 0, Eq 1). An individual j has a probability of $p_r(i)$ to be replaced by the offspring (Eq 2). N is the population size.

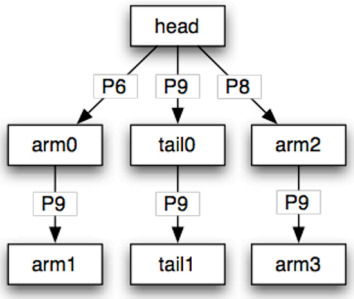
$$p_s(i) = (N - r(i)) / \sum(r(i) + 1) \quad (1)$$

$$p_r(i) = r(i) / \sum r(i) \quad (2)$$

Locomotion has been evolved with a very simple but effective fitness function. The fitness of a robot is defined as its distance from the starting position after a constant time of simulation. Therefore, the best strategy is to move in a straight line. Experience has shown that the time of simulation is crucial to achieve good results. If it is too long, the GA gets very slow but if it is too short, we only reward a fast jump at the beginning of simulation.

```
STRUCTURE
head
P6 (W arm0 W arm1)
P8 (W arm2 W arm3)
tail0 tail1

PARAMETERS
arm0.initAngle(30)
arm2.initAngle(30)
```



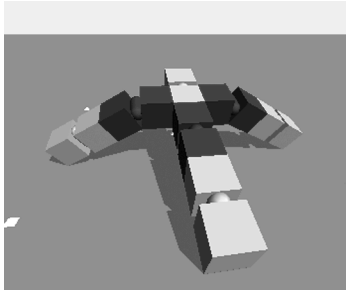


Figure 2: The script (on the left) and the corresponding genotype (in the middle). Modules are represented by nodes that encapsulate the parameters. The built structure in the simulation is on the right.

4. Results

To measure the quality of evolved individuals, we compared them with a caterpillar and a quadruped robot (figure 3) that we created with the script. To our surprise, evolved robots were not only fitter but also much more creative than our designs. For example, the *sideway roller* of figure 5 virtually tangles itself up into a knot and produces a fast rolling motion by stretching itself. The *two-legged walker* has short legs and limbs on the side that keep him from falling over. Other strategies included for example jumping, ratcheting and caterpillar like locomotion.

Simple but efficient solutions, like the first example of figure 4 were often found within the first 10 iterations of the GA. In general, highly specialized robots emerged after less than 300 iterations, but they proved to be very difficult to improve on. For example, the sideway roller of figure 5 was found already at iteration 282 but no fitter individual could be evolved afterwards.

5. Conclusions and Future Work

The Adam project has only just started but it has already proved to be a promising environment to experiment with modular robots. To the best of our knowledge, Adam is the first project that explicitly co-evolves configuration of homogenous chain-type robots.

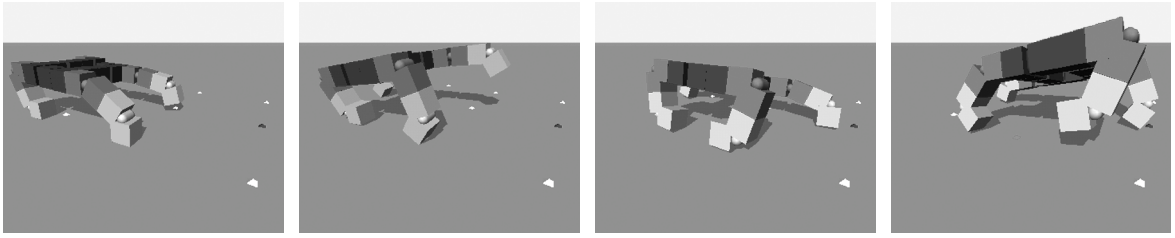
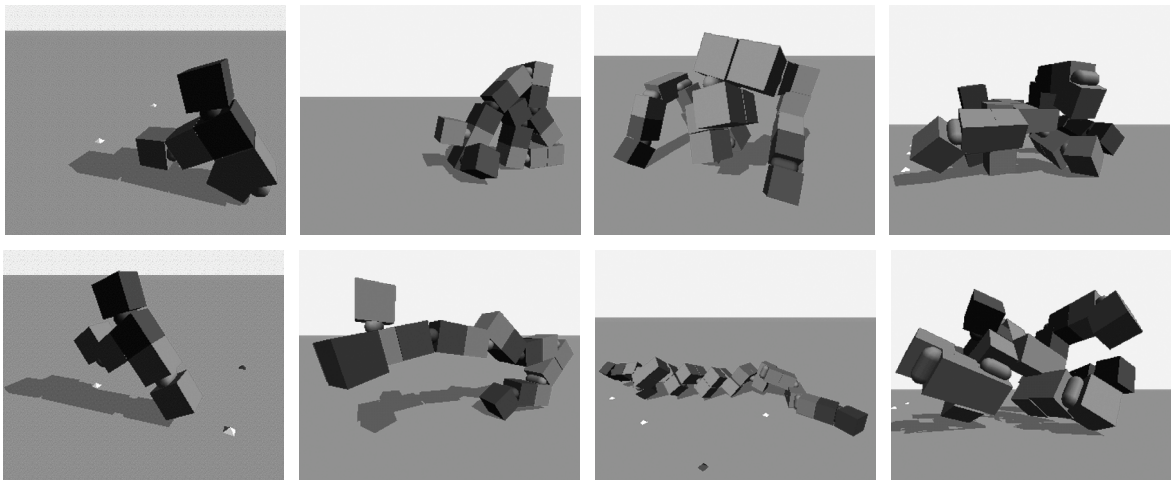


Figure 3: A simple quadruped robot that was built with the script. Modules with a rigid joint are represented in black, powered hinges in grey and elastic ones in white.



Simple jumper *Jumping worm* *Sideway roller* *Two-legged walker*

Figure 4: Locomotion strategies of evolved robots. Modules with a rigid joint are black and powered hinges are grey. Movies are available on the Adam web page at: *http://birg.epfl.ch/page32031.html*

By using modules that contain a joint and have flat connection surfaces rather than stick-like body segments that are connected through the joints, we take a step away from artificial life and one towards modular robotics. Our results so far indicate that modular chain robots are well suited for co-evolution of configuration and control. The main advantage of using only one pre-defined module type is, that evolved robots could easily be built with corresponding hardware modules.

The Adam script has proved to be a useful tool to build modular robots and to inspect evolved individuals. Scripts are easy to write and analyze. The oscillator controller is a perfect choice within this context because there are few parameters and they are easy to interpret.

In chapter 2 we presented an overview of research involving co-evolution of morphology and control. We confirm the results and observations of these projects. By co-evolving morphology and control, efficient and creative solutions are discovered. Evolved creatures display a wide range of locomotion strategies, often similar to those of living organisms in nature. Furthermore, successful individuals tend to be symmetric and redundant (e.g. sideway roller of figure 4), even though this is not directly promoted in the code. While symmetry obviously facilitates moving in a straight line, redundancy might reduce the number of fatal mutations.

However, the Adam project has only just started and its limitations are clearly apparent. Self-reconfiguration is not yet supported and the tree-based genotype does not allow the robots to have cycles and doesn't support modularity. Inspired by [15], we are currently working on a generative genotype using L-systems to evolve more complex and structured robots.

We also intend to use Adam to explore issues in locomotion control, in particular by taking inspiration of the concept of central pattern generators (CPGs), i.e. neural networks capable of producing complex patterns of oscillatory outputs without oscillatory inputs, found in vertebrate animals. As illustrated in [27, 28], CPGs can be designed as distributed systems of coupled neural or nonlinear oscillators, and produce very robust locomotion with speed, direction, and even types of gaits that can quickly be modified depending on the environmental conditions. Designing a good CPG-based controller amounts to defining the right couplings between the different oscillators and between the oscillators and the mechanical elements (e.g. in order to incorporate sensory feedback). Using the POE framework, we will explore how these couplings can be optimized in a self-organizing manner.

References

[1] E. Sanchez, D. Mange, M. Sipper, M. Tomassini, A. Perez-Uribe, A. Stauffer. Phylogeny, Ontogeny, and Epigenesis: Three Sources of Biological Inspiration for Softening Hardware. In *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, LNCS, v. 1259, Springer-Verlag, Berlin, 1997.

[2] G. Tempesti, D. Roggen, E. Sanchez, Y. Thoma. A POEtic Architecture for Bio-Inspired Hardware. In *Proc. 8th Intl. Conf. on the Simulation and Synthesis of Living Systems (Artificial Life VIII)*, Sydney, Australia, 9-13 Dec. 2002. MIT Press, Cambridge, MA, 2002, pp. 111-115.

[3] A. Kamimura, S. Murata, E. Yoshida, H. Kurokawa, K. Tomita, S. Kokaji. Self-Reconfigurable Modular Robot - Experiments on Reconfiguration and Locomotion -. In *Proceedings of 2001 IEEE/RSJ International* Conference on Intelligent Robots and Systems (IROS2001), Hawaii, USA, 2001, pp.590-597.

[4] D. Duff, M. Yim, K. Roufas. Evolution of PolyBot: A Modular Reconfigurable Robot. In *Proc. of the Harmonic Drive Intl. Symposium*, Nagano, Japan, Nov. 2001, and *Proc. of COE/Super-Mechano-Systems Workshop*, Tokyo, Japan, Nov. 2001.

[5] K. Støy, W.-M. Shen, P. Will. How to Make a Self-Reconfigurable Robot Run. In *Proceedings of the 1st international joint conference on autonomous agents and multiagent systems (AAMAS'02)*, Bologna, Italy, July 15-19, 2002.

[6] S. Vassilvitskii, J. Kubica, E. Rieffel, J. Suh, M. Yim. On the General Reconfiguration Problem for Expanding Cube Style Modular Robots. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2002.
[7] M. Vona, D. Rus. A Physical Implementation of the Self-reconfigurable Crystalline Robot. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation 2000*, San Francisco, CA, Apr. 24-28, 2000, p. 1726-1733.
[8] C. Unsal, H. Kiliccote, M. Patton, P. Khosla. Motion Planning for a Modular Self-Reconfiguring Robotic System. *Distributed Autonomous Robotic Systems 4*, Springer, November, 2000.
[9] Kohji Tomita, Satoshi Murata, Haruhisa Kurokawa, Eiichi Yoshida, Shigeru Kokaji. A Self-Assembly and Self-Repair Method for a Distributed Mechanical System. *IEEE Transactions on Robotics and Automation*, Vol.15, No.6, pp.1035-1045, 1999.
[10] E.H. Østergaard, H.H. Lund. Evolving Control for Modular Robotic Unit. In *Proceedings of CIRA'03, IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Kobe, Japan, p. 886-892, July 16-20, 2003.
[11] F. Mondada, A. Guignard, A. Colot, D. Floreano, J.-L. Deneubourg, L.M.Gambardella, S. Nolfi, M. Dorigo. SWARM-BOT: A New Concept of Robust All-Terrain Mobile Robotic System. *Technical report*, LSA2 - I2S - STI, Swiss Federal Institute of Technology, Lausanne, Switzerland, March 2002.
[12] K. Sims. Evolving 3D Morphology and Behavior by Competition. *Artificial Life IV Proceedings*, ed. by R. Brooks & P. Maes, MIT Press, p. 28-39, 1994.
[13] J. Ventrella. Explorations in the emergence of morphology and locomotion behavior in animated characters. In R. Brooks and P. Maes, editors, *Proceedings of the Fourth Workshop on Artificial Life*, Boston, MA, MIT Press, 1994.
[14] M. Komosinsky, A. Rotaru-Varga. Comparison of Different Genotype Encodings for Simulated 3D Agents. *Artificial Life Journal,* 7:395-418, 2001.
[15] G. S. Hornby, J. B. Pollack. Body-brain coevolution using l-systems as a generative encoding. In *Genetic and Evolutionary Computation Conference*, 2001.
[16] J.C. Bongard, R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Genetic and Evolutionary Computation Conference*, p. 829-836, 2001.
[17] H. Lipson, J.B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974-978, 2000.
[18] H. H. Lund. Co-evolving Control and Morphology with LEGO Robots. In Hara and Pfeifer (eds.) *Morpho-functional Machines*, Springer-Verlag, Heidelberg, 2001.
[19] H.H. Lund, R.L. Larsen, E.H. Østergaard. Distributed control in self-reconfigurable robots. In *Proceedings of ICES, The 5th International Conference on Evolvable Systems: From Biology to Hardware*, Trondheim, Norway, Springer-Verlag, March 2003.
[20] Eiichi Yoshida, Satoshi Murata, Akiya Kamimura, Kohji Tomita, Haruhisa Kurokawa and Shigeru Kokaji. Evolutionary Motion Synthesis for a Modular Robot using Genetic Algorithm. *Journal of Robotics and Mechatronics*, Vol.15, No.2, pp.227-237, 2003.
[21] J. Suthakorn A. B. Cushing G. S. Chirikjian. An Autonomous Self-Replicating Robotic System. In *Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2003.
[22] R. Nagpal, A. Kondacs, C. Chang. Programming Methodology for Biologically-Inspired Self-Assembling Systems. *AAAI Spring Symposium on Computational Synthesis: From Basic Building Blocks to High Level Functionality*, March 2003.
[23] M. Yim, J. Lamping, E. Mao, J. G. Chase. Rhombic Dodecahedron Shape for Self-Assembling Robots, Xerox PARC SPL *TechReport* P9710777, 1997.
[24] J. Urzelai, D. Floreano, M. Dorigo, M. Colombetti. Incremental Robot Shaping. In J. Koza et al. (Eds.), 3rd International Conference on Genetic Programming, San Mateo, CA: Morgan Kaufmann. In press, 1998.
[25] W. Hordijk. Population flow on fitness landscapes. *PhD thesis*, University of Rotterdam, 1994.
[26] T. Fukuda and T. Ueyama. *Cellular robotics and micro-robotics systems.* World Scientific Series in Robotics and Intelligent Systems, Vol 10. World Scientific Publishing, 1994.
[27] Ijspeert A.J., Hallam J. and Willshaw D.: Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology, *Adaptive Behavior* 7:2, pp 151-172, 1999.
[28] Ijspeert A.J.: A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander, *Biological Cybernetics*, Vol. 84:5, pp 331-348, 2001.
[29] Russell Smith. Open Dynamics Engine (ODE), *web page*: http://q12.org/ode/