Cooperative Co-evolution of Configuration and Control for Modular Robots

Chourouk Guettas, Foudil Cherif, Thomas Breton, Yves Duthen

Cooperative Co-evolution of Configuration and Control for Modular Robots

Chourouk Guettas*
LESIA Laboratory
Biskra University
Biskra, Algeria
g.chourouk@gmail.com

Foudil Cherif*
LESIA Laboratory
Biskra University
Biskra, Algeria
foud_cherif@yahoo.fr

Thomas Breton*
Irit
University of Toulouse 1
Toulouse, France
Thomas.Breton@irit.fr

Yves Duthen*
Irit
University of Toulouse 1
Toulouse, France
Yves.Duthen@ut-capitole.fr

***Abstract*—The general approach in modular robots is to hand design the morphology, and then optimizes the controller of the structure for a given task. Evolutionary robotics has proposed evolution as a bio-inspired approach to overcome the limitations of human intuition in designing robots; theoretically, the resulting structures will be better adapted. In this work, we propose an approach based on cooperative co-evolutionary genetic algorithms to design configurations and controllers for homogenous robots implicitly to support self-reconfiguration, The algorithm introduces some elements to make finding solutions easier and faster by co-evolving two populations; a population of motions sequence to search a sequence of movements that can rearrange a given modular configuration into a new one that suits a different task defined by its desired function and a population of homogenous fixed topology ANNs for the controllers to perform locomotion as a behavior evolved using genetic algorithm based on standard deviation norm. The modular robots are evaluated in a simulation environment implemented with NVidia physics engine; PhysX. The experiments carried out in this work show that co-evolving both the configuration and the controllers positively contributes to the robot's performance and optimizes its locomotion behavior.**

Keywords*—modular robots, co-evolution, controllers, artificial neural networks, genetic algorithms, locomotion.

I. Introduction

A Modular Self-reconfigurable Robot (MSR) can be considered as a universal robot because it has the ability to simulate any other physical robot, simply by connecting its modules in many different ways [1]. These modules can rearrange their connectivity which enables the self-reconfigurable robot to adapt its shape autonomously to suit any given task.

In order to operate in unstructured environments, robots will need to be adaptive; they must exhibit intelligent behavior. Evolutionary robotics, a bio-inspired method in which evolutionary algorithms are employed to optimize the control system of a robot, has been proved to overcome the limitations of human intuition in designing control strategies for autonomous machines [2]. However, the majority of work has only optimized control strategies for a human designed or bio-inspired robot morphology. This methodology has severe limitations: fixing a robots morphology places limits on the kinds of actions that the robot can perform and even if they find the optima, the resulting robot is certainly not the best possible solution for its task [3].

Nevertheless, nature manifests co-evolution; there is no distinction between a body and a brain, each doing its job. Evolution would not produce morphology and then evolve the controller. The solidarity of natures solutions emerges from co-evolutionary processes, mainly to allow creatures to survive in their environments. Co-evolution may be either cooperative between the body and the brain, or competitive between two species or with a species and its environment.

Modular robots are especially suitable to co-evolution. The complexity of designing the configuration and then programming the controller for a robot grows exponentially with the number of used modules, and might be impossible for a robot of thousands modules. Co-evolutionary algorithms give us the opportunity to evolve morphology and controllers simultaneously to suit a specific task.

II. Related work

Karl Sims [5] was a pioneer in evolving both morphology and controller for artificial creatures. The morphology and the neural systems are both genetically specified, and the morphology and behavior can adapt to each other as they evolve simultaneously. The genotypes are represented as directed graphs of nodes and connections. When simulated evolutions are performed with populations of competing creatures, diverse morphologies and gaits emerged.

Hornby and Pollack [6] used a co-evolutionary approach to evolve robots. Elementary building blocks included bars and actuators for morphology, encoded using L-systems and artificial neurons for the controller. The best creatures were used after in creating real robots.

Marbach and Ijspeert, also worked on co-evolution of configuration and controllers for modular robots. They used a tree based representation to encode configurations and controllers for Adam [7] to develop a modular robot simulation and evolution tool based on co-evolutionary approach. Using modules that contain a joint and have flat connective surfaces, body segments were connected through the joints; 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 even though this was not directly coded.

Auerbach and Bongard [2] evolve morphologies and control policies of simulated robots in virtual environments, morphologies are composed of a number of triangular meshes that

can model arbitrary shapes and thus allow for the creation of more complex morphologies than it is possible with cuboids or spheres. They used Compositional Pattern Producing Networks (CPPNs) as a generative encoding for the purpose of simultaneously evolving robot morphology and control. A method is given for translating CPPNs into complete robots including their physical topologies, sensor placements, and neural network controllers. It is shown that this method can evolve robots for a particular task.

Recently, Faina et al. [3], has proposed a co-evolutionary approach for encoding, evaluating or transferring to reality using heterogeneous modular structures with distributed control. They used a constructive evolutionary algorithm based on tree-like representations of the morphology. The algorithm introduces some new elements to smooth the search space and make finding solutions quiete easier. The evaluation of the individuals is carried out in simulations and then transferred to real robots assembled from the modules considered. All these issues are analyzed by means of an evolutionary design system called EDHMoR (Evolutionary Designer of Heterogeneous Modular Robots) that contains all the elements involved in this process. To show evidences of the conclusions of this work, EDHMoR was tested over two benchmark problems in modular robotics. The first one is focused on solving a linear robot motion mission and the second one on a static task of the robot (painting) that does not require displacements.

Like these previous studies, this current work also aims to evolve robot configurations and controllers in virtual environment. Although our approach takes inspiration from these other studies, the methods employed are distinct. The most important distinction is the type of genomic encoding utilized and using the cooperative co-evolution method [8] for evolving two populations of motion sequences and distributed homogenous neural controllers.

III. METHODS

Co-evolution can optimize the controller while developing the structure; in this way, it adapts the controller to different configurations of the robots. There is a high variety of shapes and controllers that can be developed if co-evolution is employed. throughout this research, locomotion is employed as the required task.

Our goal is to achieve a realistic simulation of homogenous modular robots; therefore this is modular robotics and not artificial life work. 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 six connections and seven degrees of freedom and we evolve the configuration and controllers of the robot using this predefined module type.

A. The proposed Modular Robot

The module has a cubic form with six connection surfaces. The form and description are inspired from Molecubes [9]. Each module is composed of two halves with an internal rotation axis defined by two diagonally opposite corners and six additional rotation axes on each connection surface (Fig. 1). Each degree of freedom has a rotational range between $[-120°, 120°]$. The dimensions of these modules are 20x20x20 mm.

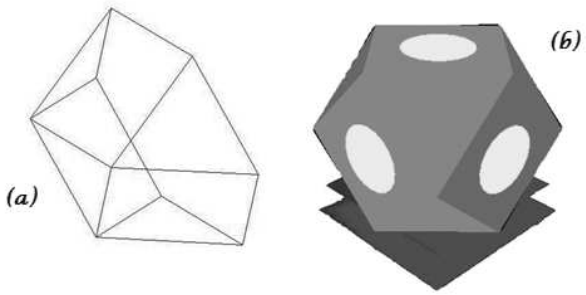

Fig. 1. (a). Polyhedron representing half module. (b). The proposed modular robot.

B. Robot's Configuration and reconfiguration

This work is based on a research done by Duthen et al. [10], which aimed to search a sequence of movements that can rearrange a given modular configuration into a new one that suits a different task. The target configuration is not explicitly specified, it is defined only by its desired function. They used Genetic Algorithms to find the optimal sequence of movements to reconfigure modules to suit another task.

The modular robots are chain-type and therefore a movement is allowed only if it keeps the modules connectivity. The connectivity of the robot is represented in a graphical format. The evolved solution is a sequence of movements executed one after another to rearrange the robots configuration.

Individuals are a sequence of movements executed one after another to rearrange the robots configuration. Each solution (individual) has four main attributes:

- *Code*: the movement is a rotation or connection / disconnection action.
- *Connector*: that will perform the action.
- *Angle*: the rotation angle, between $-120°$ and $120°$.
- *Module*: identifier of the module that will realize the action.

An initial population is generated randomly and evolved according to the parameters in Table. I , using a standard genetic algorithm.

The simulation environment was implemented using PhysX, which is widely used for video games and rigid body simulators. The results of this work were very promising and have been tested on many configurations, but its main shortcoming is that there were no controllers in the modules; therefore the tested fitness function was the greatest height possible that can be achieved by any given configuration (Fig. 2).

TABLE I. PARAMETERS USED FOR EVOLVING CONFIGURATION'S POPULATION.

Parameters	Value
Population size	60
generations	100
Mutation rate	15%
Elite population retained	40%

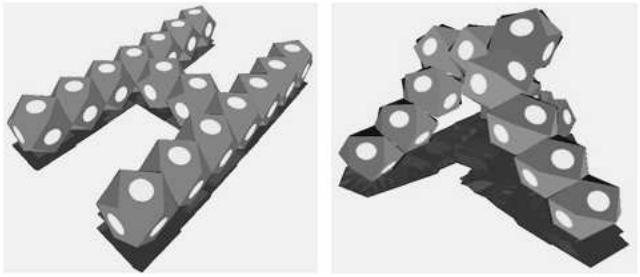

Fig. 2. One of the tested configurations tried to stand on four legs to achieve the most possible height.

We used the simulator developed in [10], and implemented our model of ANN controllers and the co-evoltionary approach on it and tested it using different configurations.

C. Controllers

In our work, we dot the cubic modules with distributed and homogeneous neural controllers. We will investigate their efficiency for locomotion based on oscillatory output from simple ANNs, starting with robotic bio-inspired configurations. We will use genetic algorithms for optimizing the synaptic weights of the ANN to produce rhythmic gaits that maximize the covered distance by the robot for predefined number of time steps.

We have implemented a fully connected feed-forward MLP ANN model (Fig. 3) with three input neurons, two output neurons, and a single hidden layer with five hidden neurons. We have tried to keep the architecture of the ANN simple, as we wanted to focus on how locomotion gaits will evolve using different configurations.

- *Input neuron 1*: This neuron represents the connector information of a module; how many neighbors are connected to this module. This neuron is fed with input between 1 and 6.
- *Input neuron 2*: This neuron is fed with the previous actuator value between $-120°$ and $120°$. These values are chosen based on many experiments.
- *Input neuron 3*: This neuron is fed with module's identification; there is no specific interval to this value because it changes according to the number of modules in different configurations.
- *Output neuron 1*: represents the actuation angle and this value is scaled to be between $-120°$ and $120°$.
- *Output neuron 2*: generates the identification of the connector that will execute the actuation, this movement will be performed only if the connector is connected.

Each module in a given configuration has its own ANN model whose output neuron is connected to the module's actuators, making it a distributed controller. Furthermore, all the modules in any given configuration have the same ANN architecture, with exactly the same topology and weight vector, making it a homogeneous distributed controller. Although all the modules have an identical neural model, the difference in their behavior emerges based on the difference in the input fed to their respective ANN at any given cycle.

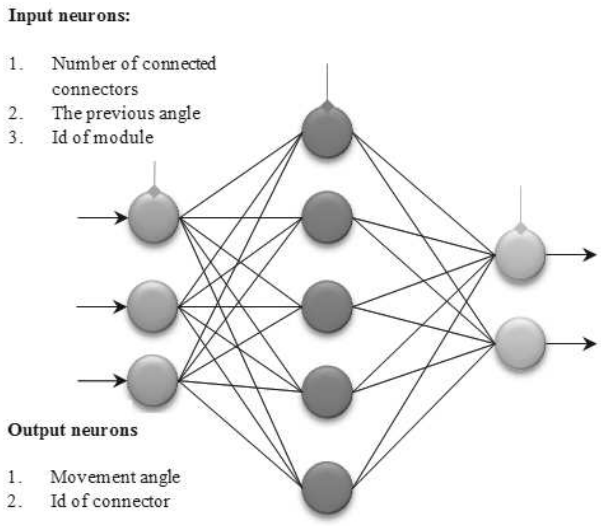

Fig. 3. A schematic of the proposed neural architecture.

With CPGs (Central Pattern Generators), if the configuration of the modular robotic changes, the coupling of the CPGs have to be changed to adapt to the new configuration, which might be a drawback if the controller is implemented on self-reconfigurable modular robots [11]. The proposed ANN model has the potential to be extended for controlling more than just locomotion for different configurations, along with the ability to increase the complexity of the hidden layer of the ANN architecture.

We have used the open source Neural Network library called Flood [12] for implementing the ANN model for the neural controller.

D. Evolving Controllers

For evolving neural controllers, we have implemented an adjusted version of genetic algorithm. By taking this path, we want to quickly achieve well-suited controllers for any given configuration.

We started with a population of random individuals, and followed the process in Fig. 4 to create new offspring. From each generation we select the best individual and after calculating its mean and standard deviation we generate a new population based on them and finally perform mutation on the new population to keep the diversity of individuals. The parameters used for simulation are illustrated in Table. II.

TABLE II. PARAMETERS USED FOR EVOLVING NEURAL CONTROLLERS.

Parameters	Value
Population size	60
generations	100
Mutation rate	1/Size of genome
Number of time-steps	100

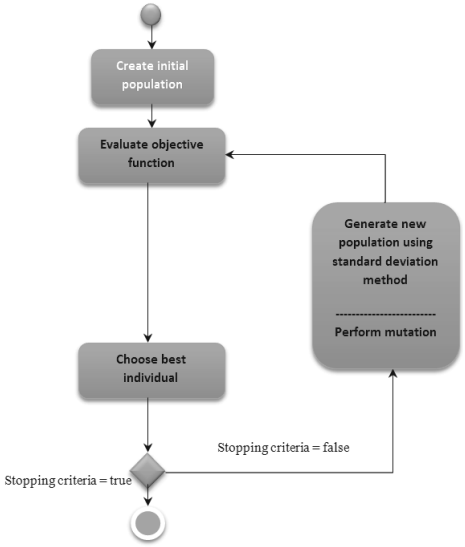

Fig. 4. Process of evolving controllers.

IV. Cooperative co-evolution

To evolve configuration and controllers, we will implement an adjusted version of a cooperative co-evolutionary algorithm presented by Mitchell Potter and Kenneth De Jong [8] to achieve well-suited configurations and controllers for any given task and environment.

Since this is preliminary work, we tested our approach in a plain environment; we started the simulation by creating the environment, and then generating two initial populations of random individuals of motions sequence and ANN controllers. To co-evolve these two populations we followed this process for each population (controllers / motions sequence) (Fig. 5):

While (not end of population)

- Create a robot;
- Choose a random individual from the other population;
- Execute the motions sequence to reconfigure the robot;
- generate the movements for n time steps using the ANN controllers;
- Evaluate the two individuals and assign their fitness (the covered distance by the robot in cm), only if it is bigger than its current fitness (if it has been already evaluated).

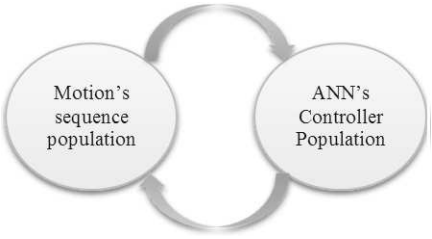

Fig. 5. Process of co-evolving motion's sequence and controllers populations.

After that, we generate new two populations and evaluate them using the same process, untill we reach the fixed number of generation.

V. Experiments and results

We experimentally tested our approach on three different configurations; we wanted to test the validity of the proposed model for evolving more adapted configurations and producing locomotive oscillations.

For each configuration, we run two tests; the first one by implementing ANN controllers on a fixed configuration, for n time steps, inputs were fed to the ANN of each module, outputs were calculated, then after scaling the output value appropriately to the range of the actuator, each module performed its actuation and finally we calculate the covered distance. For the second test, we implement the co-evolutionary approach, and evaluated it by the travelled distance as well.

A conducted comparison between the fixed configurations and the ones evolved using co-evolution will be done to show the differences between the two methods.

A. Experiment 1

For the first test, we chose the most simple and intuitive form which is the worm-like configuration, the modular robots consists of five modules and we run the simulation twice; evolving just locomotion (Fig. 6.a), co-evolving configuration and controllers (Fig. 6.b).

Fig. 6.c plots the covered distance by the best performing individuals of the two approaches; evolving locomotion (20.96 cm) and co-evolving both controllers and configuration (54.48 cm). The neural controller was able converge and settle into a stable oscillation pattern and into a stable locomotion gait within a very short period of time for both approaches.

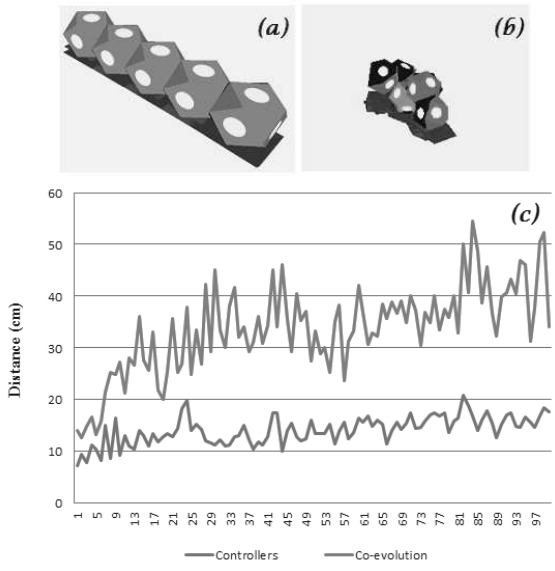

Fig. 6. (a). 5 modules' robot (b). A co-evolved configuration (c). Distance covered by the best performing individuals of each approach.

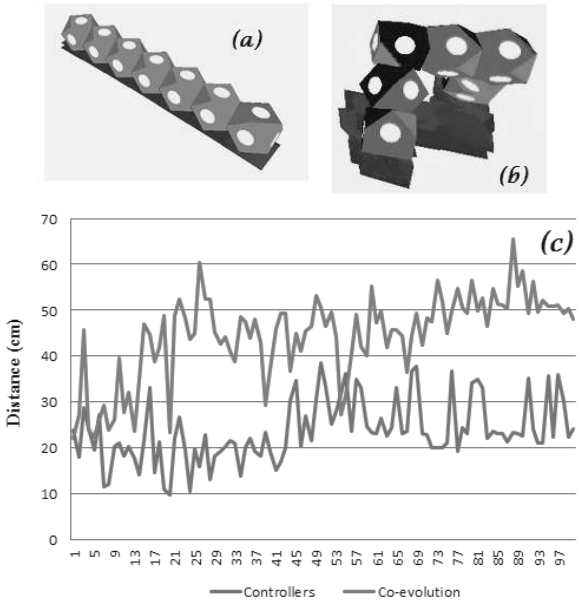

Fig. 7. (a). 7 modules' robot (b). A co-evolved configuration (c). Distance covered by the best performing individuals of each approach.

We validated the same model by evolving seven module linear configurations (Fig. 7.a, Fig. 7.b), which produced a locomotion gait with a covered distance of 37.45 cm for just locomotion, and for co-evolution 65.48 cm (Fig. 7.c).

B. Experiment 2

For the second test, we implemented our model on a four-legged configuration, Fig. 8.a, 8.b, under the same conditions of the first test.

The results as shown in Fig. 8.c which plots the covered distance by the best performing individuals of the two approaches; evolving locomotion (42.56 cm) and co-evolving both controllers and configuration (69.64 cm). As it was observed in the first experiment the controller quickly converged and settled into stable oscillation pattern.

C. Experiment 3

As for the last experiment, we tested the model on a spiderlike configuration, Fig. 9.a, Fig. 9.b, and we ran tests according to the evolution process.

Fig. 9.c plots the difference between the covered distances by the best individual of each approach; the one of evolved locomotion was able to travel for 31.80 cm, and the other one covered a distance 94.58 cm. There was a quick converge and settle into stable different oscillation patterns for both tests.

VI. Discussion

All evolved individuals reconfigure at least by changing an angle between two modules, but not all of them were able to move significantly far from the starting position.

Simulations time varies between the different configurations, and gradually increases with the complexity of the configuration.

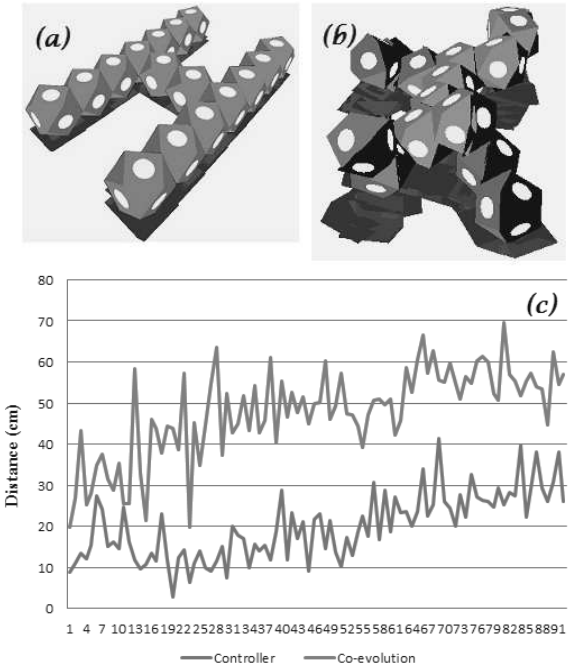

Fig. 8. a. Four-legged robot (16 modules). b. A co-evolved configuration c. Distance covered by the best performing individuals of each approach.

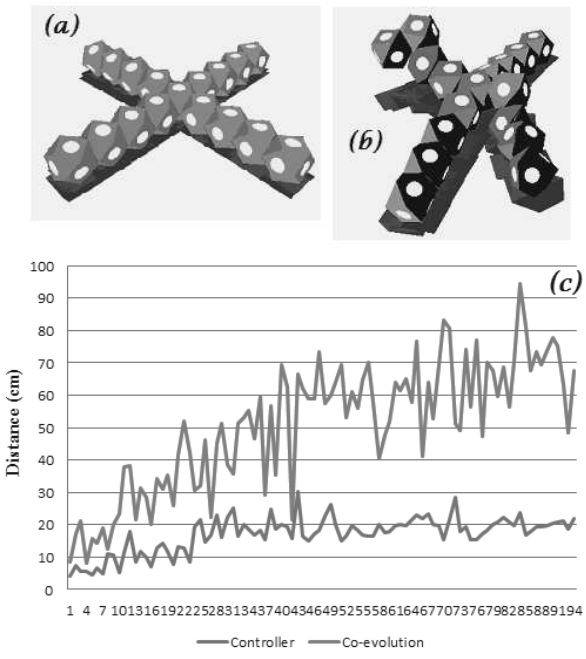

Fig. 9. (a). Spider-like robot (17 modules) (b). A co-evolved configuration (c). Distance covered by the best performing individuals of each approach.

There are several important observations based on the results:

- The neural controller is able to quickly converge and settle into a stable oscillation pattern and with that into a stable locomotion gait for all the configurations, using simple ANN model to investigate the minimal

neural architecture required for locomotion. However, some produced locomotion strategies for complex configurations could cover large distances but a detailed examination showed that they received high impacts from the ground. These high impacts are not tolerable by the real hardware of modular robots, since the modules might break.

- A variety of locomotion gaits were evolved, while a few of them seem as if they were symmetrical.
- The instability of evolution curves is due to the used process in generating new populations. This instability does not prevent the production of more effective gaits, since the evolution curve is gradually increasing.
- By implementing a cooperative co-evolutionary approach we show that evolving both morphology and control realize increasingly better behaviors for modular robots and increases the rate of adaptation by covering larger distance. This was demonstrated for three different configurations which travelled the maximum distance (Fig. 10).
- Sometimes minor changes in configuration generate better behaviors, which does prove that modular robots have their own characteristics which must be taken into consideration while designing robots.
- The characteristics of the proposed modular robot restrict the configurations' search space, which prevents the evolution of better performing configurations.

VII. Conclusion and perspectives

The work has only just started but it has already proved to be promising in producing locomotive oscillations for different configurations and in co-evolving more adapted modular robots. To the best of our knowledge, this is the first work that implicitly co-evolves configuration and controllers of homogenous chain-type robots to support self-reconfiguration even though the proposed modular robot imposes much contraints on the configurations' search space.

For the next first step, we would like to focus on evolving both the topology and the weight of the ANN using the NEAT methodology; to both increase the complexity of the architecture for adaptive locomotion and other behaviors in different robotic configurations and to evolve modular robots in more complex environments rather than simple ones, to achieve more accurate results on multiple levels and approach real life scenarios. Secondly, To propose an hybrid modular robot model that support self-reconfiguration to evolve robots for different tasks.

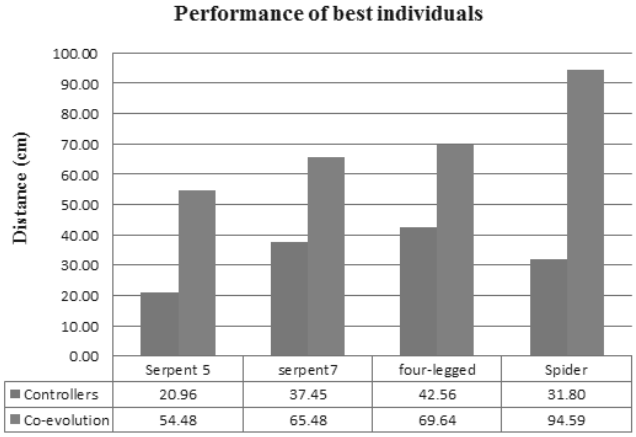

Fig. 10. Best performing individuals in each approach.

Acknowledgment

We specially thanks Yves Duthen, for facilitating our research by providing us with his developed modular robotics simulator and all the necessary documentations.

References

[1] K. Stoy, D. Brandt and D. J. Christensen, *Self-Reconfigurable Robots: An Introduction*, Cambridge, MA: MIT Press, 2010.

[2] J. E. Auerbach and J. C. Bongard, "On the relationship between environmental and morphological complexity in evolved robots," in *Proc. Genetic and Evolutionary Computation Conf.*, 2012, pp. 521-528.

[3] A. Faina, F. Bellas, F. Lpez-Pena and R. J. Duro, "EDHMoR: Evolutionary Designer of Heterogeneous Modular Robots," in *Engineering Applications of Artificial Intelligence*, vol. 26, no. 10, pp. 2408-2423, November 2013.

[4] J. C. Bongard, "How evolution shapes the way roboticists think," *Procedia Computer Science*, vol. 7, pp. 8-10, 2011.

[5] K. Sims, "Evolving 3D Morphology and Behavior by Competition," in *Proc. Artificial Life IV*, R. Brooks and P. Maes, ed. Cambridge, MA: MIT Press, 1994, pp. 28-39.

[6] G. S. Hornby and J. B. Pollack, "Body-Brain Co-evolution Using L-systems as a Generative Encoding," in *Proc. Genetic and Evolutionary Computation Conf.*, 2001, pp. 974-978.

[7] D. Marbach and A. J. Ijspeert, "Co-evolution of configuration and control for homogenous modular robots," in *Proc. 8th conf. on Intelligent Autonomous Systems*, 2004, pp. 712-719.

[8] M. A. Potter and K. A. De Jong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1-29, 2000.

[9] V. Zykov, W. Phelps, N. Lassabe and H. Lipson, "Molecubes Extended : Diversifying Capabilities of Open-Source Modular Robotics", in *International Conf. Intelligent RObots and Systems*, 2008.

[10] Y. Duthen, H. Luga, N. Lassabe, S. Cussat-Blanc, T. Breton and J. Pascalie, "An introduction to the Bio-Logic of Artificial Creatures," *Intelligent Computer Graphics*, vol. 321, pp 1-23, 2010.

[11] A. Ranganath, J. Gonzalez-Gomez, and L. M. Lorente, "Distributed Neural Controller for Locomotion in Linear Modular Robotic Configurations," *Book chapter (VII)*, Centro de automtica y Robtica CSIC-UPM, pp. 129-144, 2011.

[12] R. Lopez, 2010, *Flood: An Open Source Neural Networks C++ Library* [Online]. Available: http://www.cimne.com/flood.