Online Gait Learning for Modular Robots with Arbitrary Shapes and Sizes

Massimiliano D'Angelo[1], Berend Weel[2,3], and A.E. Eiben[3]

[1] University La Sapienza, Rome, Italy
[2] University of York, UK
[3] VU University Amsterdam, The Netherlands
maxxi.d.angelo@gmail.com, {b.weel,a.e.eiben}@vu.nl

Abstract. This paper addresses a principal problem of *in vivo* evolution of modular multi-cellular robots. To evolve robot morphologies and controllers in real-space and real-time we need a generic learning mechanism that enables arbitrary modular shapes to obtain a suitable gait quickly after 'birth'. In this study we investigate a reinforcement learning method and conduct simulation experiments using robot morphologies with different size and complexity. The experiments give insights into the online dynamics of gait learning, the distribution of lucky / unlucky runs and their dependence on the size and complexity of the modular robotic organisms.

Keywords: embodied artificial evolution, modular robots, artificial life, online gait learning, reinforcement learning.

1 Introduction

The work described in this paper forms a stepping stone towards the grand vision of embodied artificial evolution (EAE) as outlined in [6]. The essence of this vision is to construct physical systems that undergo evolution 'in the wild', i.e. not in a virtual world inside a computer. There are various possible approaches towards this goal including chemical and biological ones. The one behind this study is based on using a mechatronical substrate, that is, robots.

In general, there are two principal forces behind evolution: selection and reproduction. Selection –at least environmental, objective-free selection– is 'for free' in the real world. Therefore, the main challenge for EAE is reproduction, i.e., the creation of tangible physical artifacts with the ability to reproduce. In our case, this means the need for self-reproducing robots. The approach we follow to this end is based on modular robotics with robotic building blocks capable of autonomous locomotion and aggregation into complex 'multicellular' structures in 3D. This approach has several advantages. Firstly, it offers a high level of control because the basic modules or cells are robots themselves that can be predesigned and preprogrammed according to the preferences of the experimenter. Here one can choose a homogeneous system with identical building blocks or a heterogeneous one with several kinds of modules. Secondly, modular robotics offers a high level of flexibility, because the number of different combinations, that is, different aggregated 3D structures, is huge. This means that the design space of all possible aggregated robotic organisms is rich enough to accommodate interesting evolutionary processes.

It is important to note that the morphology of the basic robot modules is fixed by design and cannot change during the operational / experimental period. Hence, evolution will not take place in the morphological space of these pre-engineered modules, but in the morphological space of the multicellular organisms. From the perspective of the multicellular robot bodies the basic robots are merely raw material whose physical properties do not change over time.[1]

Recently these ideas have been put on a more solid footing by presenting a conceptual framework for systems where robot morphologies and controllers can evolve in real-time and real-space [5]. This framework, dubbed the Triangle of Life, describes a life cycle that does not run from birth to death, but from conception (being conceived) to conception (conceiving one or more children) and it is repeated over and over again, thus creating consecutive generations of 'robot children'. The Triangle of Life consists of 3 stages, Birth, Infancy, and Mature Life.

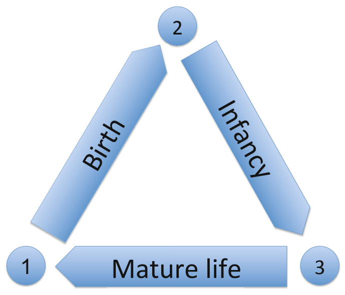


Fig. 1. The Triangle of Life. The pivotal moments that span the triangle and separate the 3 stages are: 1) Conception: A new genome is activated, construction of a new organism starts. 2) Delivery: Construction of the new organism is completed. 3) Fertility: The organism becomes ready to conceive offspring.

In this paper we address a fundamental problem in the Infancy stage. This stage starts when the morphogenesis of a new robot organism is completed and the 'baby robot' is delivered. As explained in [5], the body (morphological structure) and the mind (controller) of such a new organism will unlikely fit each other well. Even if the parents had well matching bodies and minds, recombination and mutation can easily result in a child where this is not the case. Hence, the new organism needs some fine tuning; not unlike a newborn calf the 'baby robot' needs to learn how to control its own body. This problem –the Control

[1] Nevertheless, evolving the controllers of these elementary robot modules during the operational period is possible.

Your Own Body (CYOB) problem– is inherent to evolutionary ALife systems where both bodies and minds undergo changes during reproduction.

The work described here addresses the general CYOB problem in a simplified form, by reducing it to gait learning. In the modular robots approach the challenge is to find a method that can learn gaits for all different morphologies that can be created with the given modules and can do this quickly. The problem is highly nontrivial, since a modular robot organism has many degrees of freedom, which leads to a very large search space of possible gaits. Furthermore, this learning process must take place on-the-fly, during the real operational period of the robot organisms. The off-line approach, where a good controller is developed (evolved, learned, hand-coded, ...) before the robot is deployed is not applicable here, because the life cycle of the Triangle is running in a hands-free mode without being paused for intervention by the experimenter.

The mechanism we employ here to solve the CYOB problem is reinforcement learning, in particular, the PoWER algorithm described by Kober and Peters [14]. Note, that this learning mechanism is not evolutionary itself. Evolution takes place on the level of multicellular robot organisms (as it is these organisms that reproduce and get selected in the Triangle of Life framework), whereas the PoWER algorithm is applied inside one organism to discover a good controller that induces a good gait. The grand evolutionary process is not investigated here; it only forms the background context that raises the CYOB problem. The specific research questions our experiments will try to answer are the following:

1. How is the run-time dynamics of the learning process? That is, how quickly can a multi-cellular robot organism learn to walk?
2. How reliable is the learning process? That is, how often do we see lucky (unlucky) runs resulting in fast walking (immobilized) robot organisms?
3. How do the above features depend on the size and complexity of the robot morphologies?

2 Related Work

The design of locomotion for modular robotics is a difficult problem. As explained by Spröwitz: Locomotion requires the creation of rhythmic patterns which satisfy multiple constraints: generating forward motion, without falling over, with low energy, possibly coping with different environments, hardware failures, changes in the environment and/or of the organism [18]. In the literature there are several approaches, based on various types of controllers and algorithms for creating these rhythmic patterns.

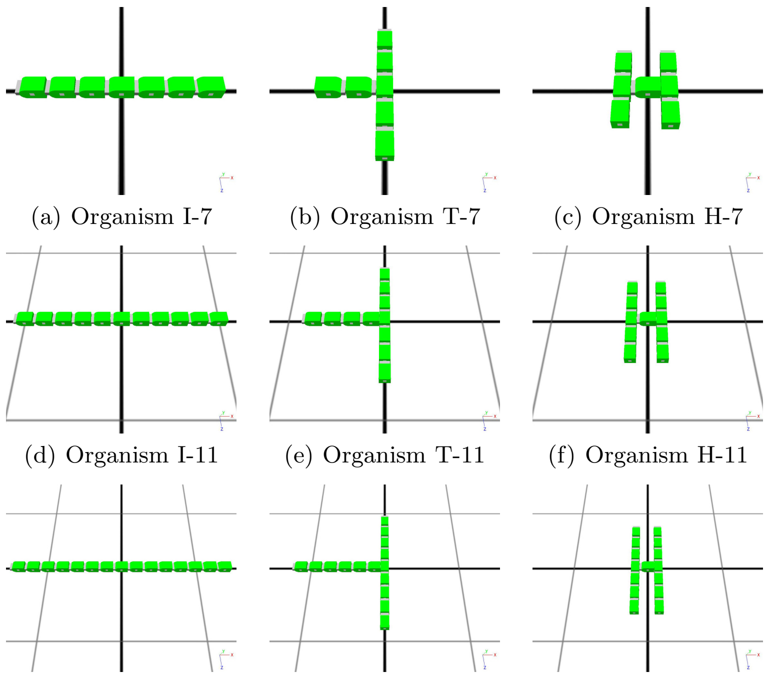
One of the earliest types is gait control tables as in, for instance, [1] and [19]. A gait control table consist of rows of actuator commands with one column for each actuator, each row also has a condition for the transition to the next row, in essence this implements a very simple cyclic finite state machine. A second major avenue of research is that of neural networks (NN). In particular for locomotion of robot organisms HyperNEAT is used extensively. HyperNEAT is an indirect encoding for a neural network. The genome is a compositional pattern producing network (CPPN), these networks are directed graphs in which each node is a mathematical function like sine, cosine or a Gaussian [4]. CPPN's are used to set the weights of a fixed size neural network called a substrate. Several studies have shown that HyperNEAT is capable of creating efficient gaits for robots [4,20,9]. Another successful approach that has received much attention is based on Central Pattern Generators (CPG). CPGs model neural circuitry found in vertebrates which output cyclic patterns without requiring a cyclic input [11]. Each actuator in a robot organism is controlled by the output of a CPG, furthermore the CPGs are connected through certain variables which allows them to synchronise and maintain a certain phase difference pattern. Although sensory input is not strictly needed for CPG's, it can be incorporated to shape the locomotion pattern to allow for turning and modulating the speed. This technique has been shown to produce well performing and stable gaits on both non-modular robots [18,2] and modular multi-robot organisms [13,12]. Last, a technique based on artificial hormones has been investigated for the locomotion of modular robot organisms. In this technique artificial hormones are created within robot modules as a response to sensory inputs. These hormones can interact with each other, diffuse to neighbouring modules and act upon output hormones. These output hormones are then used to drive the actuators [10,17]. Furthermore, some techniques in the field of gait learning employ reinforcement learning algorithms, the specific approaches used can range from Temporal Difference Learning (TDL) to Expectation-Maximization (EM). In TDL one seeks to minimize an error function between estimated and empirical results of a controller, in EM controller parameters are estimated in order to maximize the reward gained using it. These algorithms have been used on modular, e.g. [3] and non modular robots, e.g. [16].

Although there is extensive previous work on this issue, we must stress that, of the techniques described above, only the techniques described in [3], [12] and [18] were actually tested on multiple shapes.

3 Experimental Setup

As mentioned in Section 1 our primary goal is to determine if a reinforcement learning (RL) approach can be suitable for online gait learning. To this end, we tested an RL algorithm in various organism morphologies (with different sizes and complexity) set in a simple environment. All these tests were done in simulation with the Webots system of Cyberbotics, using the YaMoR module as building block for the organisms [15]. A YaMoR module is made of a static body and a joint on its front that has a single degree of freedom and an operating range of $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$. It also has two connectors, one on the joint and one in the back of the body, which allow to connect modules at arbitrary angles. For the current investigation we applied three changes to the original YaMoR model. We added two extra connectors on the remaining sides of the body in a central position, allowing the construction of complex structures. We reduced the width of the joint to avoid its lateral protrusion, thereby eliminating the possibility of collisions with the modules connected to the sides. Last, we added a GPS device to a centrally located module to measure displacement.

Nine different robot organisms with different sizes and complexity were defined so as to examine the algorithm generality and scalability. Size and complexity are measured by the number of modules and by the number of extremities, respectively. The experiments were conducted with three complexity levels: organisms with two extremities (I-shape), three extremities (T-shape), and four extremities (H-shape). Each shape was then replicated in three sizes: 7, 11 and 15 modules. A screenshot of their initial state can be seen in Figure 2.



(a) Organism I-7 (b) Organism T-7 (c) Organism H-7

(d) Organism I-11 (e) Organism T-11 (f) Organism H-11

(g) Organism I-15 (h) Organism T-15 (i) Organism H-15

Fig. 2. Robot Organisms

The environment chosen for the experiments is an infinite plane free of obstacles so to avoid any extra complexity and the need of supervision. Each experiment starts with the organism lying completely flat at the plane origin.

The controller of each organism defines an open-loop gait. Because we use reinforcement learning to acquire a good gait, we refer to our controllers as policies. In our implementation, a policy is a set of cyclic spline functions where each spline specifies the angular positions of an actuator over time. A cyclic spline is a mathematical function that can be defined using a set of n control points. Each control point is defined by (t_i, α_i) where t_i represents time and α_i the corresponding value. $t_i \in [0, 1]$ is defined as

$$t_i = \frac{i}{n-1}, \forall i = 0, \ldots, (n-1) \tag{1}$$

and $\alpha_i \in [0, 1]$ is freely defined, except that the last value is enforced to be equal to the first, i.e. $\alpha_0 = \alpha_n$. These control points are then used for cyclic spline interpolation using GSL [7] dedicated C functions. Using GSL it is possible to query a spline for a different number of points than it was defined with, enabling comparison between splines defined with a different number of parameters.

The problem is then to find a set of splines, that maximizes some measure of performance. For this we use the RL algorithm called PoWER described by Kober and Peters [14]. The use of the set of cyclic spline functions as the representation was taken from [16]. This RL algorithm is based on an Expectation-Maximization approach to estimate the parameters $\hat{\alpha}$ of a policy π in order to maximize the reward gained by using that policy. The algorithm starts by creating the initial policy π_0 with as many splines as there are robots (actuators). The algorithm initialises these splines with n values of 0.5 and then adding Gaussian noise. This initial policy is then evaluated after which it is adapted. This adapted controller is evaluated and adapted again until the stopping condition is reached.

Adaptation is done in two steps which are always applied: Exploitation and Exploration. In the exploitation step, the current splines $\hat{\alpha}$ are optimized based on the outcome of previous controllers, this generates a new set of splines.

$$\hat{\alpha}_{i+1} = \hat{\alpha}_i + \frac{\sum_{j=1}^{k} \hat{\Delta}\alpha_{i,j} R_j}{\sum_{j=1}^{k} R_j} \tag{2}$$

where $\hat{\Delta}\alpha_{i,j}$ represents the difference between the parameters of the i-th policy and j-th policy belonging to a ranking of the best k policies seen so far and R_j its reward. In the exploration phase policies are adapted by applying Gaussian perturbation to the newly generated policy.

$$\hat{\alpha}'_{i+1} = \hat{\alpha}_{i+1} + \hat{\varepsilon}_{i+1}, \, \hat{\varepsilon}_{i+1} \sim \mathcal{N}\left(0, \sigma^2\right) \tag{3}$$

where $\hat{\alpha}_{i+1}$ are the parameters after the exploitation step, $\hat{\alpha}'_{i+1}$ the parameters after the exploration step and $\hat{\varepsilon}_{i+1}$ values drawn from a Gaussian distribution with mean 0 and variance σ^2. Over the course of evaluations, the variance σ^2 is diminished which decreases exploration and increases exploitation.

To carry out the actual evaluation each policy is queried for its parameters at a rate proportional to the actuators' angular speed. These are then appropriately scaled depending on actuators' operating range and used cyclically during the evaluation. Each controller is evaluated for 96 seconds (6,000 time steps) after being used for a recovery period of 3.2 seconds (200 time steps) in order to reduce evaluation noisiness as in [8]. This is a rather long evaluation time for locomotion, but was chosen to be conservative and allow for a reasonably

Table 1. Experiment Parameters

Parameter	Value	Description
Variance	0.008	The initial variance
Variance Decay	0.98	The variance decay factor
Ranking Size	10	Number of best policies to compare against
Start Parameters	2	Starting number of parameters of a spline
End Parameters	100	Ending number of parameters of a spline
Evaluation Steps	6,000	Number of time steps for evaluations
Recovery Steps	200	Number of time steps for recovery
Evaluations	1,000	Number of evaluations

accurate measurement of performance. The reward R awarded to a controller i is calculated as follows:

$$R_i = \left(100\frac{\sqrt{\Delta_x^2 + \Delta_y^2}}{\Delta_t}\right)^6 \tag{4}$$

where Δ_x and Δ_y is the displacement over the x and y axes measured in meters and Δ_t the time spent in evaluation, as in [16].

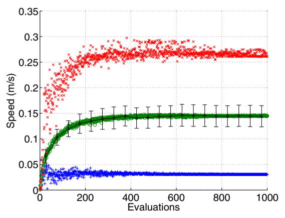
The algorithm operating parameters used for the variance and its decay factor are the same as in [16] whereas the others were chosen by hand, without further tuning. The experiment was repeated for 30 times for each organism, with different random seeds. An overview of all the parameters used in each experiment are described in Table 1.

4 Experimental Results

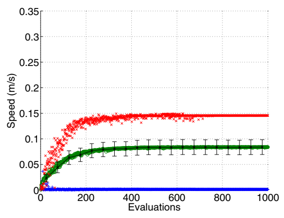
The performance of the algorithm is shown in Fig. 3, the graphs show the mean fitness of the controllers over 30 runs. We assess the algorithms' behaviour by the time it needs to converge and by the quality of the learned controllers. In Fig. 3 we can see that the algorithm converges after roughly 400 evaluations, regardless of the organisms shape or size. We also note that after convergence, the performance of the gait is very stable.

To reach this performance, a complete run took about 27 hours of simulated time, or just over 1 full day. In light of current hardware, where an operating time of over 4 hours is rare, this is still too long to be feasible in real hardware. However, as the algorithm converges in approximately one third of this time we can imagine reducing this time immensely. For instance, as mentioned in Section 3 the choice of evaluation length was conservative, it should therefore be safe to assume it is possible to reduce the evaluation time without affecting the results significantly.

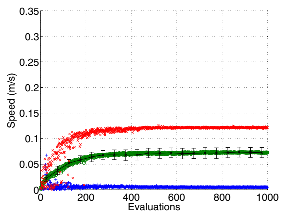
The blue curves shown in Fig. 3 display that some runs are very bad. In such runs the average speed of the organism increases until it suddenly drops



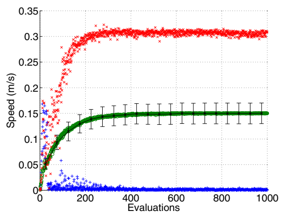
(a) Organism I-7



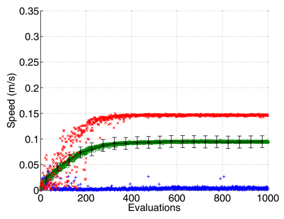
(b) Organism T-7



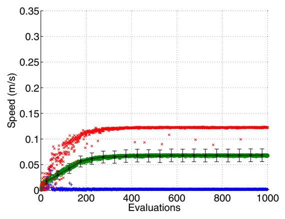
(c) Organism H-7



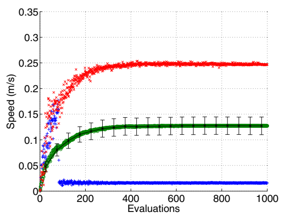
(d) Organism I-11



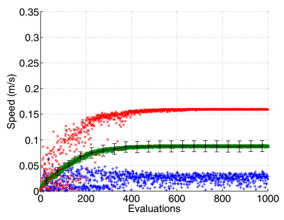
(e) Organism T-11



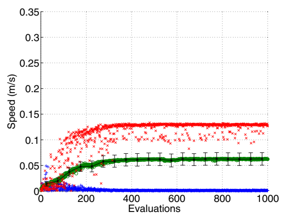
(f) Organism H-11



(g) Organism I-15



(h) Organism T-15

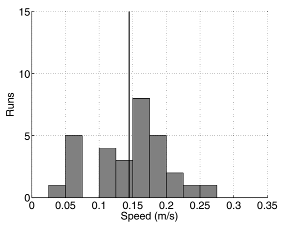


(i) Organism H-15

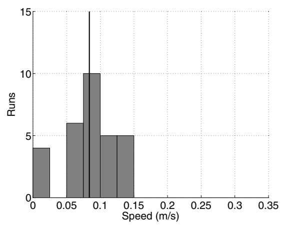
Fig. 3. Online dynamics of the learning process. The x axis represents time measured by the number of evaluations whereas the y axis represents evaluation performance measured by the average speed attained (m/s). The top (red) and bottom (blue) curves exhibit the best and worst single run out of the 30, respectively. The middle curves (green) show the average speed over 30 runs with the black bars representing the confidence interval.

below the 0.05 m/s mark. The reason for this huge drop is that a particular gait during the learning process made the organism lose its balance and flip on its side (I-shape), head (T-shape) or altogether (H-shape). From that point on the learning process was unable to find a good set of parameters for the now completely changed situation.

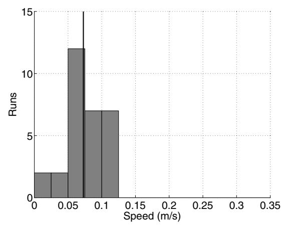
This behaviour can have two causes. First, the on-line approach used implies that between each evaluation the organism's stance or position is not reset to a default stable one. Consequently, each controller's performance is affected by the state the organism was left in by the previous one, meaning that a good move in one stance may lead to disastrous behaviour in another stance. Second, some organism morphologies may be more prone than others to lose their balance due to detrimental gaits. In [16] such detrimental gaits are filtered out based on knowledge of the organism shape and size, however, as our approach is meant



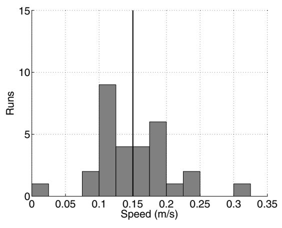
(a) Organism I-7



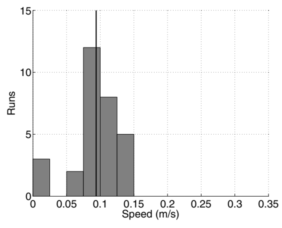
(b) Organism T-7



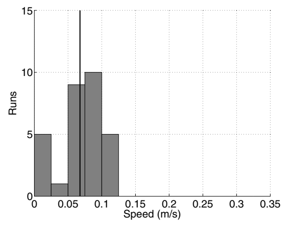
(c) Organism H-7



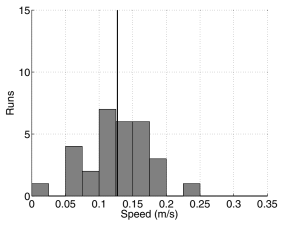
(d) Organism I-11



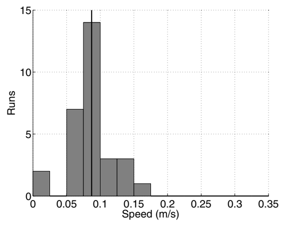
(e) Organism T-11



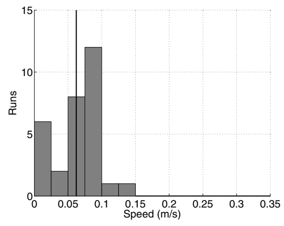
(f) Organism H-11



(g) Organism I-15



(h) Organism T-15



(i) Organism H-15

Fig. 4. Reliability of the learning process. The histograms show the number of runs (out of 30) terminating with a speed in a given range.

for on-line adaptation for arbitrary shapes and sizes, it is not possible to filter such gaits as we do not have knowledge of the size and shape beforehand.

All of these reasons, combined with the decreasing exploration, may have led to such bad runs. In the Triangle of Life, however, bad runs should not be a big problem, shapes which are prone to bad runs will not be able to reproduce and hence these shapes disappear over time. On the other hand, shapes that are well suited for balanced locomotion will be able to prevail.

The histograms in Fig. 4 show the distribution of runs based on the performance of the last controller, the black vertical bar represents the mean. The distribution of runs allows us to determine the reliability of the learning process by the number of bad runs for each organism. These histograms show quite clearly for each organism the number of runs of which the outcome is a controller with a very low average speed (below 0.05 m/s). Over the different experiments there is a minimum of 1 bad run and a maximum of 7 bad runs, which corresponds to a percentage between 3.3% and 23.3%.

Table 2. T-Test Results - Size. The table show the results of the independent samples T-Tests comparing the performance of the last controller on organisms with same complexity but different size. **NS** means the T-test showed the difference is not significant, the corresponding p values are included.

	I	T	H
7 - 11	**NS**, p = 0.710	**NS**, p = 0.300	**NS**, p = 0.531
7 - 15	**NS**, p = 0.210	**NS**, p = 0.718	**NS**, p = 0.188
11 - 15	**NS**, p = 0.094	**NS**, p = 0.423	**NS**, p = 0.537

Table 3. T-Test Results - Complexity. The tables show the results of the independent samples T-Tests comparing the performance of the last controller on organisms with same size but different complexity. **NS** means the T-test showed the difference is not significant, **S** means the difference *is* significant, the corresponding p values are included.

	7	11	15
I - T	**S**, $p = 1.602 \times 10^{-05}$	**S**, $p = 1.821 \times 10^{-05}$	**S**, $p = 2.873 \times 10^{-04}$
I - H	**S**, $p = 7.959 \times 10^{-08}$	**S**, $p = 5.472 \times 10^{-09}$	**S**, $p = 1.119 \times 10^{-07}$
H - T	**NS**, p = 0.217	**S**, p = 0.005	**S**, p = 0.004

The number of bad runs seems to be quite constant among organisms of the same complexity with different sizes, the I shape for instance has very few bad runs regardless of whether its size is 7 or 15 modules. On the other hand across complexity there does seem to be an influence on the number of bad runs: the T and H shapes have relatively more bad runs than the I shape. More experiments need to be carried out in order to assess if this relationship between the number of bad runs and organism complexity holds more generally. Going back to the performance of the organisms, Fig. 3 shows that organisms with same complexity but different size are very similar in performance. Organisms with the same size but different complexity, however, show a very large difference in performance. To see if these differences are statistically significant T-Tests with a significance level of ($p < 0.01$) were conducted using the performance of the last controller of each run. The results are shown in Tables 2 and 3. There we can see that indeed the difference in performance between organisms of the same complexity, but with different sizes is not significant. The difference in performance between organisms of the same size, but different complexity is significant in most cases. This strengthens our belief that the design space for robot organisms consisting of modular robots is an interesting one.

5 Conclusions

In this paper we addressed a principal problem of *in vivo* evolution of modular multi-cellular robots. The Control Your Own Body problem arises because newborn robot organisms are likely to have bodies and controllers that do not fit well. Therefore, every 'baby robot' needs to learn to control its own body. Furthermore, it needs to learn this quickly and on-the-fly by an online learning method, because Life goes on in the Triangle of Life, without a grace period. In this study we reduced this to a gait learning problem and investigated a solution by applying reinforcement learning. We conducted simulation experiments using robot morphologies of different size and complexity.

The main finding of our research is that the RL PoWER algorithm can successfully perform this learning task. Its success seems to depend more on the shape than on the size of the organisms, while its speed proved rather independent from either of these factors. The differences between morphologies can be quite substantial: The I shape had very few bad runs where the organism fails to move at all, whereas the the failure rate for the H shape can be up to 20%. These failures are due to very bad gaits that cause the organism to lose its balance and flip on its side or back.

Regarding the speed of the learning process we found that the learning algorithm converges to the best gait after around 400 evaluations for all shapes and sizes we tested here. In terms of time, our experiments completed in roughly 27 simulated hours for 1,000 evaluations, where each evaluation ran for 96 seconds. However, we expect that the evaluations could be made shorter without losing much performance as the evaluation time of 96 seconds is rather long compared to other gait learning algorithms. Using shorter evaluations and stopping the learning process after 400 evaluations we could reduce the learning period to approximately 2 hours simulated time.

Further work will be carried out along three lines. First of all, we want to improve the RL PoWER algorithm by tuning its parameters. Furthermore, we are interested in comparing this method to other methods, for instance HyperNEAT. Last but not least, we want to validate our results by replicating these experiments using real hardware.

References

1. Bongard, J., Zykov, V., Lipson, H.: Resilient machines through continuous self-modeling. Science 314(5802), 1118–1121 (2006)
2. Christensen, D.J., Larsen, J.C., Støy, K.: Fault-tolerant gait learning and morphology optimization of a polymorphic walking robot. Evolving Systems (to appear, 2013)
3. Christensen, D.J., Schultz, U.P., Støy, K.: A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots. Robotics and Autonomous Systems 61(9), 1021–1035 (2013)
4. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 2764–2771. IEEE Press (2009)
5. Eiben, A.E., Bredeche, N., Hoogendoorn, M., Stradner, J., Timmis, J., Tyrrell, A., Winfield, A.: The triangle of life: Evolving robots in real-time and real-space. In: Lió, P., Miglino, O., Nicosia, G., Nolfi, S., Pavone, M. (eds.) Advances in Artificial Life, ECAL 2013, pp. 1056–1063. MIT Press (2013)

6. Eiben, A.E., Kernbach, S., Haasdijk, E.: Embodied artificial evolution. Evolutionary Intelligence 5(4), 261–272 (2012)
7. Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Booth, M., Rossi, F.: Gnu Scientific Library: Reference Manual. Network Theory Ltd. (2009)
8. Haasdijk, E., Eiben, A.E., Karafotias, G.: On-line evolution of robot controllers by an encapsulated evolution strategy. In: IEEE Congress on Evolutionary Computation, CEC 2010, pp. 1–7. IEEE Press (2010)
9. Haasdijk, E., Rusu, A.A., Eiben, A.E.: HyperNEAT for locomotion control in modular robots. In: Tempesti, G., Tyrrell, A.M., Miller, J.F. (eds.) ICES 2010. LNCS, vol. 6274, pp. 169–180. Springer, Heidelberg (2010)
10. Hamann, H., Stradner, J., Schmickl, T., Crailsheim, K.: A hormone-based controller for evolutionary multi-modular robotics: From single modules to gait learning. In: IEEE Congress on Evolutionary Computation, CEC 2010, pp. 1–8. IEEE Press (2010)
11. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: A review. Neural Networks 21(4), 642–653 (2008)
12. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic locomotion design and experiments for a modular robotic system. IEEE/ASME Transactions on Mechatronics 10(3), 314–325 (2005)
13. Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Kokaji, S., Murata, S.: Distributed adaptive locomotion by a modular robotic system, M-TRAN II. In: Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2004, vol. 3, pp. 2370–2377. IEEE Press (2004)
14. Kober, J., Peters, J.: Learning motor primitives for robotics. In: IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 2112–2118. IEEE Press (2009)
15. Möckel, R., Jaquier, C., Drapel, K., Dittrich, E., Upegui, A., Ijspeert, A.: YaMoR and Bluemove – an autonomous modular robot with Bluetooth interface for exploring adaptive locomotion. In: Tokhi, M.O., Virk, G.S., Hossain, M.A. (eds.) Proceedings of the 8th International Conference on Climbing and Walking Robots, CLAWAR 2005, pp. 685–692. Springer (2006)
16. Shen, H., Yosinski, J., Kormushev, P., Caldwell, D.G., Lipson, H.: Learning fast quadruped robot gaits with the RL PoWER spline parameterization. Cybernetics and Information Technologies 12(3), 66–75 (2012)
17. Shen, W.-M., Salemi, B., Will, P.: Hormones for self-reconfigurable robots. In: Pagello, E., Groen, F., Arai, T., Dillman, R., Stentz, A. (eds.) Proceedings of the 6th International Conference on Intelligent Autonomous Systems (IAS-6), pp. 918–925. IOS Press (2000)
18. Spröwitz, A., Moeckel, R., Maye, J., Ijspeert, A.J.: Learning to move in modular robots using central pattern generators and online optimization. The International Journal of Robotics Research 27(3-4), 423–443 (2008)
19. Yim, M.: A reconfigurable modular robot with many modes of locomotion. In: Proceedings of International Conference on Advanced Mechatronics, pp. 283–288. Japan Society of Mechanical Engineers, Tokio (1993)
20. Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J., Lipson, H.: Evolving robot gaits in hardware: the HyperNEAT generative encoding vs. parameter optimization. In: Lenaerts, T., Giacobini, M., Bersini, H., Bourgine, P., Dorigo, M., Doursat, R. (eds.) Advances in Artificial Life, ECAL 2011, pp. 890–897. MIT Press (2011)