Online Gait Learning for Modular Robots with Arbitrary Shapes and Sizes

Weel, Berend; D'Angelo, M.; Haasdijk, Evert; Eiben, A. E.

published in
Artificial life
2017

DOI (link to publisher)
10.1162/ARTL_a_00223

document version
Publisher's PDF, also known as Version of record

On-line Gait Learning for Modular Robots with Arbitrary Shapes and Sizes

Berend Weel[1] M. D'Angelo[2] Evert Haasdijk[1]
A.E. Eiben[1]

[1]Computer Science Department, VU University, Amsterdam, Netherlands

[2]University La Sapienza, Rome, Italy

b.weel@vu.nl maxxi.d.angelo@gmail.com e.haasdijk@vu.nl a.e.eiben@vu.nl

Abstract

Abstract: Evolutionary robotics using real hardware is currently restricted to evolving robot controllers, but the technology for evolvable morphologies is advancing quickly. Rapid prototyping (3D-printing) and automated assembly are the main enablers of robotic systems where 'robot babies' can be produced based on a blueprint that combines the morphologies and the controllers of the parents. This paper addresses the principal problem of gait learning in newborn robots whose morphology is unknown in advance. We investigate a reinforcement learning method and conduct simulation experiments using robot morphologies with different size and complexity. We establish that reinforcement learning does the job well and it does it better than two alternatives. The experiments also give insights into the on-line dynamics of gait learning and the impact of the size, shape, and complexity of the modular robotic organisms. These insights can potentially be used to predict the viability of modular robotic organisms before they are constructed.

Keywords: *evolutionary robotics, embodied evolution, modular robots, artificial life, on-line gait learning, reinforcement learning*

Published as:
Online Gait Learning for Modular Robots with Arbitrary Shapes and Sizes.
Berend Weel, M. D'Angelo, Evert Haasdijk, and A. E. Eiben Artificial Life 2017 23:1, 80–104.
https://doi.org/10.1162/ARTL_a_00223

1 Introduction

This work forms a stepping stone towards the grand vision of the Evolution of Things as outlined in [11, 14, 12]. 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 avenues towards this goal including chemical, biological, and robotic approaches. This study falls in the latter category based on modular robots composed of elementary modules or building blocks. Here one can choose a homogeneous system with identical building blocks or a heterogeneous one with several kinds of modules. In both cases a robotic organism (or robot, for short) is an aggregated three-dimensional structure. It is these structures that form the possible robot morphologies and morphological evolution will take place in the space of all possible robotic organisms.

The algorithmic aspects of systems where robot morphologies and controllers can evolve in real-time and real-space have been put on a solid footing by a conceptual framework dubbed the Triangle of Life [13]. This framework describes the pivotal life cycle of an ecosystem of self-reproducing robots. This life cycle 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 result is a population of robotic organisms that evolves and thus adapts to the given environment. The Triangle of Life consists of 3 stages, Morphogenesis, Infancy, and Mature Life as illustrated in figure 1.

This paper focusses on learning in the Infancy stage. We assume that there is a procedure for Morphogenesis that can produce new robotic organisms and deliver these in the habitat. As explained in [13], the body (morphological structure) and the mind (controller) of a new robotic 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 robot needs to do some fine tuning; not unlike a newborn calf the 'baby robot' needs to learn how to control its own body. This problem –the Control Your Own Body (CYOB) problem– is inherent to Artificial Life systems where newborn organisms are random combinations of the bodies and minds of their parents.

This paper considers a particular, limited, instance of the general CYOB problem: gait learning. The overall goal of this paper is to find an appropriate method for gait learning across a range of different morphologies that can be created with the given modules and that can do this quickly. The problem is highly nontrivial, since a modular robotic organism has many degrees of freedom, which leads to a very large search space of possible gaits. Furthermore, this learning process must take place in an on-line fashion, during the real operational period of the robotic organisms. The off-line approach, where a good controller

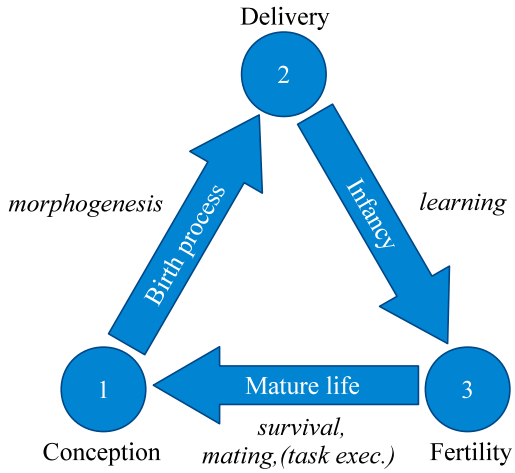


Figure 1: *The Triangle of Life. The pivotal moments that span the triangle and separate the 3 stages are: 1) Conception: A new genome is activated, construction of a new robot starts. 2) Delivery: Construction of the new robot is completed. 3) Fertility: The robot becomes ready to conceive offspring.*

is developed before the robot is deployed, is not applicable here, because the life cycle of the Triangle is running without being paused for intervention by the experimenter.

The mechanism we investigate here to solve the CYOB problem is reinforcement learning, in particular, the RL PoWER algorithm described by Kober and Peters [26]. Note, that this learning mechanism is not evolutionary itself. Evolution takes place on the level of multicellular robotic organisms: these organisms are the units of reproduction, whereas the RL PoWER algorithm is applied inside one organism as an individual learning mechanism to discover a good controller that induces a good gait. The specific questions that will be addressed in this paper are the following:

1. Is RL PoWER suitable to learn a good gait on-the-fly in 'robot babies' with arbitrary shapes and sizes? How does RL PoWER compare to other options, such as Simulated Annealing and HyperNEAT?

2. How is the learning process affected by the shapes and sizes of the robots? Can we identify morphological features that are good predictors of the quality of the learned gait?

The second question is particularly relevant for future implementations in hardware, because the creation of a physical organism is expensive. Predicting the walking abilities of

robotic organisms before they are constructed can help filter out hopeless shapes, thus making the use of system resources more efficient. This paper extends more limited experiments published in [8].

2 Related Work

Evolutionary Robotics is the combination of evolutionary computing and robotics [2, 9, 15, 28, 37, 38, 39]. The field "aims to apply evolutionary computation techniques to evolve the overall design, or controllers, or both, for real and simulated autonomous robots" [38]. This approach is "useful both for investigating the design space of robotic applications and for testing scientific hypotheses of biological mechanisms and processes" [15]. However, as noted in [2] "the use of metaheuristics [i.e., evolution] sets this subfield of robotics apart from the mainstream of robotics research" which "aims to continuously generate better behavior for a given robot, while the long-term goal of Evolutionary Robotics is to create general, robot-generating algorithms". This provides the context for our work that aims to employ on-line evolution in real-time and real-space to deliver robot morphologies and controllers suited for a given environment. As outlined in [10] such a system can be used for engineering purposes as well as for conducting fundamental studies into evolution in a novel substrate.

Currently we are not aware of any evolutionary robotic systems that implement the principles of the Triangle of Life. However, there are two studies that come close in some important aspects. Weel *et al.* proposed a "Robotic Ecosystem with Evolvable Minds and Bodies" that works with on-line evolution without a centralized evolution manager 'above' the robot population [40]. The system is a genuine implementation of the Birth, Infancy, and Mature life stages in a simulated circular habitat. Robots are not required to perform any specific task and are free to move in any direction, limited only by the border of the habitat. The robots have a modular morphology constructed from Roombots and a controller that is a set of parametrised cyclic splines describing the servo motor angles in the Roombot joints as a function of time. Body and mind are encoded by a genome and there are appropriate crossover and mutation operators to create new genomes from given parents. Learning, in particular of gaits, in the Infancy stage is implemented through reinforcement learning.

Recently, Brodbeck *et al.* published an experimental study "Morphological Evolution of Physical Robots through Model-Free Phenotype Development" [4]. The overall objective is to demonstrate a "model-free implementation for the artificial evolution of physical systems, to stochastically optimize the design of real-world machines". Being model-free means that the system does not employ simulations, all robots are physically constructed. Again, the system is based on modular robot morphologies. Two types of cubic modules (active and passive) form the 'raw material' and robot bodies are constructed from a handful of such

modules. The robots do not have their own 'brain', i.e., their own on-board controller. The robots' actions are determined by a central controller running on a desktop PC that sends the commands for the active modules via Bluetooth to the microcontrollers that operate the servos. The task is locomotion, which is achieved by oscillating the servos at a certain frequency and amplitude determined by the genome. The evolutionary process is conducted by a centralized evolutionary algorithm that runs on the external PC. The fitness is the travelled distance in a given time interval. The robot phenotypes are constructed in real hardware for fitness evaluation and the system was designed to construct new robots autonomously.

These papers represent important milestones towards the Evolution of Things. They demonstrate the feasibility of such systems in a complimentary manner. The first paper demonstrates a full ecosystem: a robot population with evolvable morphologies and controllers, where 'parents' select each other autonomously and 'babies' undergo learning in real time — but only in simulation. Even though the use of an existing hardware platform (Roombots) makes the system constructible, it has not been constructed in the real world. The second paper showcases a genuine hardware implementation, where the robotic manipulator ('mother robot') and the given supply of modules form a Birth Clinic. However, evolution is an off-line, centrally managed process, where robots are built and tested in isolation (not as part of a living population) and the robots are driven by commands from an external PC.

Simulated and real world experiments have been contrasted before in (evolutionary) robotics. This led to the notion of the reality gap, the effect that the working of an evolved controller or morphological feature obtained in simulations will not be the same once transferred to real hardware [23]. The rationale for using simulations for this paper is twofold. First, while our longer term goal is a learning mechanism for real robots, this is an exploratory stage of development where simulations offer a relatively quick[1] method to assess and compare different options. Second, the reality gap concerns the difference between simulated or real world behaviour of robot controllers, whereas here we are comparing learning mechanisms. We believe that our analyses of performance patterns and the conclusions about ranking the three learning methods (RL Power, Simulated Annealing, and HyperNEAT) will also stand in real hardware.

The design of locomotion for modular robots is a particular problem. It amounts to the creation of rhythmic patterns which satisfy multiple requirements: generating forward motion, without falling over, with low energy usage, possibly coping with different environments, hardware failures, changes in the environment and/or the organism [35]. In the

[1]The quickness of simulations is indeed relative. For instance, the tests with the 270 random robot shapes took about 15 days of computing time for *each learning mechanism* on a 12 core i7 Mac Pro with the Webots simulator.

literature there are several approaches, based on various types of controllers for creating these rhythmic patterns as well as various algorithms to optimise their parameters.

One of the earliest approaches recognizes the periodicity of these patterns and successfully exploits this by utilizing cyclic genetic algorithms to evolve gaits [29, 30]. Another classic approach is based on gait control tables as in, for instance, [3] and [41]. 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, essentially creating a very simple cyclic finite state machine.

A second major avenue of research considers evolving controllers based on neural networks. HyperNEAT [36] provides a particularly popular approach to evolving robot controllers, in particular for locomotion tasks. HyperNEAT's indirect encoding for neural networks separates the genotype, a compositional pattern producing network (CPPN) from the phenotype, the "subtrate". CPPNs are directed graphs with weighted edges in which each node is a mathematical function like sine, cosine or Gaussian (similar to neural networks but with a variety of activation functions). The evolved CPPNs specify the weights of the substrate, the neural network that actually controls the robot. Several studies have shown that HyperNEAT is capable of creating efficient gaits for modular and for legged robots [7, 42, 19].

Another successful approach that has received much attention is based on Central Pattern Generators (CPG). CPGs model neural circuitry found in vertebrates that outputs cyclic patterns without requiring a cyclic input [22]. Each actuator in a robotic organism is controlled by the output of a CPG, and the CPGs are connected to allow them to synchronise and maintain a certain phase difference pattern. Although sensory input is not strictly required 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 [5] and modular multi-robotic organisms [35, 25, 24].

Recently, a technique based on artificial hormones has been investigated for the locomotion of modular robotic 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 [20, 33].

Furthermore, there are gait learning techniques that employ reinforcement learning algorithms; the specific approaches used can range from Temporal Difference Learning to Expectation-Maximization. Temporal Difference Learning seeks to minimize an error function between estimated and empirical results of a controller. Expectation Maximization estimates controller parameters so as to maximize the reward gained. These algorithms have

been used on modular (e.g., [6]) as well as non modular robots (e.g., [32]).

Although there is extensive previous work on this issue, we must stress that, of the techniques described above, only the ones in [6], [24] and [35] were actually tested on multiple shapes. In this category, the most closely related work is the recent paper of Rossi and Eiben who use a similar test suite of twelve modular robots and on-line evolution running on every robot [31].

3 Experimental Setup

As mentioned in section 1 our primary goal is to assess the suitability of a particular reinforcement learning approach called RL PoWER for on-line gait learning. To this end, we test the RL PoWER algorithm on various hand designed organism morphologies with different sizes and complexity. To obtain a relative assessment, we compare the performance of RL PoWER to on-line implementations of Simulated Annealing and HyperNEAT. Furthermore, we apply a powerful off-line optimisation method, CMA-ES, with a high computational budget to obtain a good indication of the practical maximum of robot speeds. This provides benchmark results to position the (on-line) learning performance of RL PoWER more precisely.

Our second goal is to investigate in which way the morphology of an organism influences the efficacy of the learning process. We apply the RL PoWER method on a large number of randomly generated morphologies, and characterize their shapes in terms of traits such as size, number of extremities, number of effective degrees of freedom (DoF) and effective DoF per extremity. An experimental analysis shows which of these traits provide good predictors for the learned gait.

3.1 Robot morphologies

All experiments were carried out in simulation with the Webots simulator (Cyberbotics), using the YaMoR module as the building block for the organisms [27]. A YaMoR module (see figure 2) 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. The original

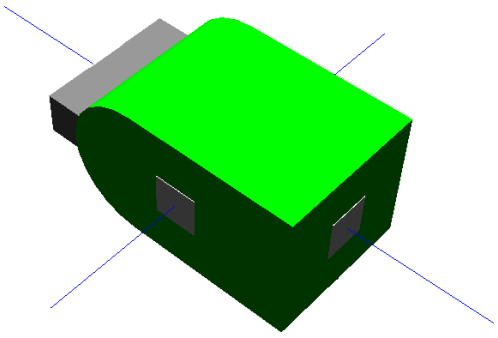


Figure 2: *Modified YaMoR module. The blue lines show the perpendicular axis of the connectors*

YaMoR model was modified in three ways to accommodate the current experiments. We added two extra connectors on the left and right sides of the body in a central position, allowing the construction of more complex morphologies. We reduced the width of the joint by removing its lateral protrusions, thus eliminating the possibility of collisions with modules connected to the sides. Lastly, we added a global positioning sensor to a centrally located module to enable exact measurements of the organism's to locomotive success.

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 experiments for RL PoWER, Simulated Annealing and HyperNEAT were repeated for 30 times for each organism, with different random seeds.

When comparing the performance levels achieved with learners, this paper compares the speed achieved in the final evaluation of each run. Some runs may be unlucky, then, because that last evaluation just happens to consider a poor candidate controller. Thus, the measure provides an accurate reflection of the *actual* performance of the robots during on-line learning, putting learners with erratic performance across evaluations (e.g., because they consider many poor controllers as well as good ones) are at a disadvantage.

3.2 Robot controllers

The controller used with RL PoWER, Simulated Annealing and CMA-ES is a set of cyclic splines that define an open-loop gait. Each spline within this set, or policy, specifies the angular positions of a single actuator over a certain amount of time. A cyclic spline is a mathematical function that is 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

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

and $\alpha_i \in [0, 1]$ is freely defined. An additional control point (t_n, α_n) is defined to enforce that

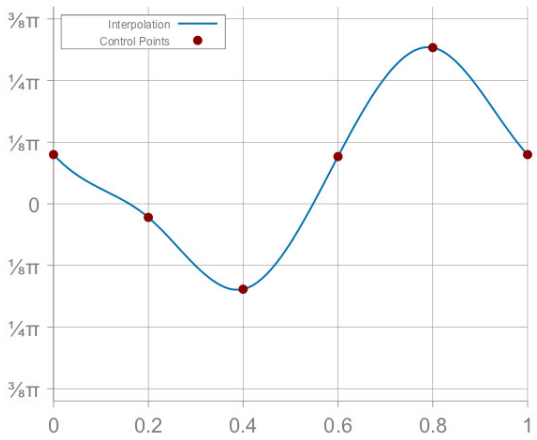


Figure 3: *Example of a spline interpolated from 6 control points*

the last value is equal to the first, i.e. $\alpha_0 = \alpha_n$
and so enable cyclic splines. These control points
are then used to interpolate a cubic spline with periodic boundary conditions using GSL [16] dedicated C functions. Using GSL it is possible to query a spline for a different number of points than it was defined with, an example is shown in Figure 3. This use of sets of cyclic spline functions as the representation was taken from [32]. The task of the learning algorithm is then to optimise the parameters of a set of splines so that performance –distance travelled, in this case– is maximised. Each controller is evaluated for 23.76 seconds (1,485 time steps). Before each evaluation starts, the controller runs for a recovery period of 3.168 seconds (198 time steps) to reduce evaluation noise as suggested in [18].

The HyperNEAT experiments use neural net based controllers that are described in section 3.5.

3.3 RL PoWER

The RL PoWER algorithm was proposed by Kober and Peters [26]. This reinforcement learning algorithm is based on an Expectation-Maximization approach to estimate the parameters of a policy to maximise the reward gained.

The algorithm creates the initial policy with as many splines as there are modules, each having 2 control points. These control points are initialised at 0.5 and then perturbed using Gaussian noise. The algorithm then enters an evaluation-adaptation loop to refine the policy, until the stopping condition is reached. A ranking of k best policies encountered so far is kept to inform the adaptation of the current policy.

Adaptation consists of three components: spline size increase, exploitation and exploration. The spline is gradually refined by incrementing the number of control points periodically as proposed in [32] In the exploitation step, the current parameters are adapted based on the values of the k best policies. In the exploration phase policies are adapted by applying Gaussian perturbation to the policy resulting from exploitation. Over the course of the run the variance σ^2 is diminished which decreases exploration and increases exploitation.. The pseudocode for the algorithm, as defined in [26], is displayed in Algorithm 1.

The reward awarded to a controller is calculated as:

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

where Δ_x and Δ_y is the displacement over the x and y axes measured in meters and Δ_t the evaluation time.

```
initialisation;
policy ← intialisation;
evaluate(policy);
while evaluation < total_evaluations do
    /* Update the ranking of k best policies                    */
    ranking.insert(policy);
    if ranking.size > k then
        ranking.remove_worst;
    end
    /* Spline size increase                                     */
    if evaluation mod increase_delta = 0 then
        spline_size ← spline_size + 1;
        reinterpolate_all(ranking);
        reinterpolate(policy);
    end
    /* Exploitation                                             */
    rewards ← 0;
    weighted_total ← 0;
    for p in ranking do
        rewards ← rewards + p.reward;
        weighted_parameters ← p.reward * (policy.parameters - p.parameters);
        weighted_total ← weighted_total + weighted_parameters;
    end
    next_policy.parameters ← policy.parameters + weighted_total / (rewards + ϵ);
    /* Exploration                                              */
    next_policy.parameters ← next_policy.parameters + normrnd(0,sqrt(variance));
    policy ← next_policy;
    variance ← variance * variance_decay;
    evaluate(policy);
end
```

Algorithm 1: RL PoWER

The operating parameters for RL PoWER, such as the variance and its decay factor, as well as the reward function, were taken from [32]. The values were: 0.008 for the variance and 0.98 for the variance decay, k is set to 10. The splines are initialised with 2 control points and are allowed to grow to a maximum of 100 control points over the course of a run, in this case the spline is grown every 10 evaluations ($round(\frac{1000}{100-2} = 10)$. ϵ is a parameter to avoid division by 0 and is set to 10^{-10}.

3.4 Simulated Annealing

The second algorithm we use is the well known Simulated Annealing algorithm [1]. The pseudocode of our implementation can be seen in Algorithm 2. The initialisation of the of the first policy is the same as for RL PoWER, e.g. the initial policy π_0 is created with as many splines as there are modules. The algorithm initialises these splines with n values of 0.5 and then adds Gaussian noise, for this algorithm we chose $n = 20$. The temperature is initialised at 1,000, the cooling-rate and σ are set to 0.1. Cooling rate and σ values were selected after a limited parameter sweep.The spline size was chosen heuristically to be large enough to allow fine-grained splines, while keeping the dimensionality in check.

$$P(a, b) = \begin{cases} 1 & \text{if } b > a \\ e^{-\frac{a-b}{T}} & \text{otherwise} \end{cases} \tag{3}$$

$$E = 1000 \frac{\sqrt{\Delta_x^2 + \Delta_y^2}}{\Delta_t} \tag{4}$$

where Δ_x and Δ_y is the displacement over the x and y axes measured in meters and Δ_t the evaluation time. The Simulated Annealing algorithm uses Equation 3 as the acceptance function and Equation 4 as the energy function.

3.5 HyperNEAT

HyperNEAT offers a popular and competitive neuroevolutionary approach to robot locomotion which is typically used in off-line applications [7]. HyperNEAT evolves a neural network's connectivity pattern indirectly, using a generative encoding called a Compositional Pattern Producing Network (CPPN). Like a neural network, a CPPN is a network of mathematical functions with weighted connections. Unlike neural networks, the network can contain a variety of activation functions including Sine, Cosine, Gaussian and Sigmoid. To determine the weight of a connection in the neural network that controls the robot (the

```
begin
    α̂_current ← intialization;
    T ← initialization;
    evaluate α̂_current;
    for i in evaluations do
        α̂_next = α̂_current + N(0, σ);
        evaluate α̂_next;
        if P(E(α̂_current), E(α̂_next)) > random(0, 1) then
            α̂_current ← α̂_next;
        end
        T ← T * (1 − coolingRate);
    end
end
```

Algorithm 2: Simulated Annealing

substrate), the coordinates of the two substrate nodes are fed into the CPPN which then returns the connection weight [36].

The HyperNEAT experiments do not use a spline-based controller, but a neural network substrate that controls a closed-loop gait and is composed of three layers: input, hidden and output layer. Each layer is an $m \times n$ matrix of nodes where $m = (OrganismSize_x * 2) - 1$ and $n = (OrganismSize_y * 2)$, with $OrganismSize_x$ and $OrganismSize_y$ the sizes of the organism respectively on the x and y axes measured by the number of modules. The inputs are the angular positions of each module's servo at the previous time step together with a sinusoidal signal $s = \sin(\omega t)$ where ω represents the maximum angular velocity of the modules servo and t the current time. The network outputs the angular positions of each module servo for the current time step. The input and output signals are scaled to and from the interval $[-1, +1]$. The population size is 25, further HyperNEAT settings have the standard values as found in J. Gauci's implementation, on which we based our code[17].[2]

Just as for the RL PoWER and Simulated Annealing experiments, this paper considers the results of running HyperNEAT in an on-line setting with the evaluations in series on the robot, without resetting or repositioning the robot between evaluations.

[2] `https://github.com/MisterTea/HyperNEAT`

3.6 Practical Optimum

To have a good indication of the practical maximum of attainable robot speeds, we generate benchmark controllers with an established powerful solver for complex numerical optimisation problems, CMA-ES [21]. In contrast to the algorithms we investigate, CMA-ES runs off-line. We use standard settings and a computational budget of 4,000 evaluations per run for each organism. (The on-line learners have a budget of 1,000 evaluations.) The listed practical optimum for each shape is the best controller obtained over five replicate CMA-ES runs for that shape.

3.7 Test Suite I: Designed Shapes

To assess the applicability of the RL PoWER algorithm for gait learning, we test it on nine designed robotic organisms with different sizes and shapes as shown in figure 4. We compare the RL PoWER results to those of Simulated Annealing and HyperNEAT.

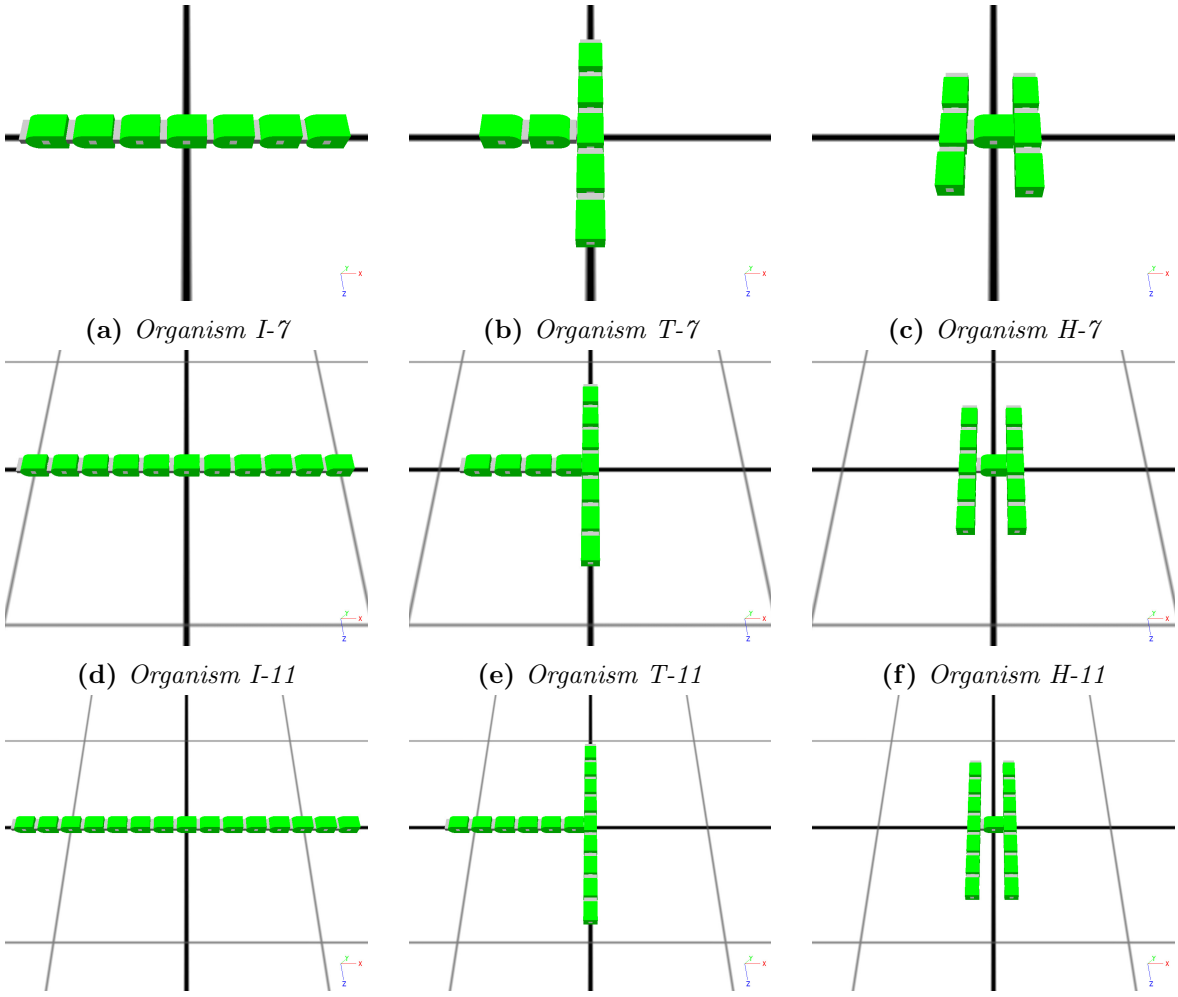
The size and complexity of the organisms are measured by the number of modules and by the number of extremities, respectively. The experiments were conducted with three complexity levels: organisms with two extremities (I-shape), three extremities (T-shape), and four extremities (H-shape). We created each shape in three sizes: 7, 11 and 15 modules. Figure 4 shows simulation screenshots of the shapes in their initial, prone, state.

3.8 Test Suite II: Randomly Generated Shapes

The context of a system where the robots' shapes evolve requires a learning algorithms that performs well on arbitrary shapes. We recreate this context by creating a number of shapes randomly and performing 30 replicate runs of the RL PoWER algorithm on each shape. This allows us to gauge RL PoWER's efficacy also on shapes that may not be very practicable for locomotion. Secondly, it enables us to investigate the impact of shape and size of an organism further than we could with the limited set of designed shapes.

The second test suite was constructed by generating 270 random shapes as follows[3]. The size of the organism is determined by drawing a random number between 3 and 15 (uniform distribution). Construction of the organism then starts on the basis of a single root module. The organism is extended to the target size by selecting a random open connector on the organism and attaching a module to that connector with orientation randomly selected as one of 0, 90, 180 or 270 degrees. This process was repeated to generate 270 unique shapes (duplicate shapes were discarded and regenerated).

[3]Pictures of all shapes and movies of the gaits they learned are available at
`http://www.few.vu.nl/~bwl400/rlpower/`



(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*

Figure 4: *Our first test suite of designed robotic organisms*

Because the YaMoR module is not symmetrical and the side connectors are not completely centred, the resulting shapes can have conflicts in the module placements. These conflicts have been resolved manually by rotating one of the conflicting modules. In most cases this repair did not result in a functionally different shape. In some cases the conflicts had to be resolved through multiple rotations or a rotation that did result in a functionally different shape. For the purpose of these investigations such functionally different shapes can still be considered randomly generated[4].

The designed shapes were characterised in terms of size and the number of extremities, for the generated shapes we consider two additional metrics: the number of effective joints and the number of effective joints divided by the number of extremities. These metrics are explained in more detail below.

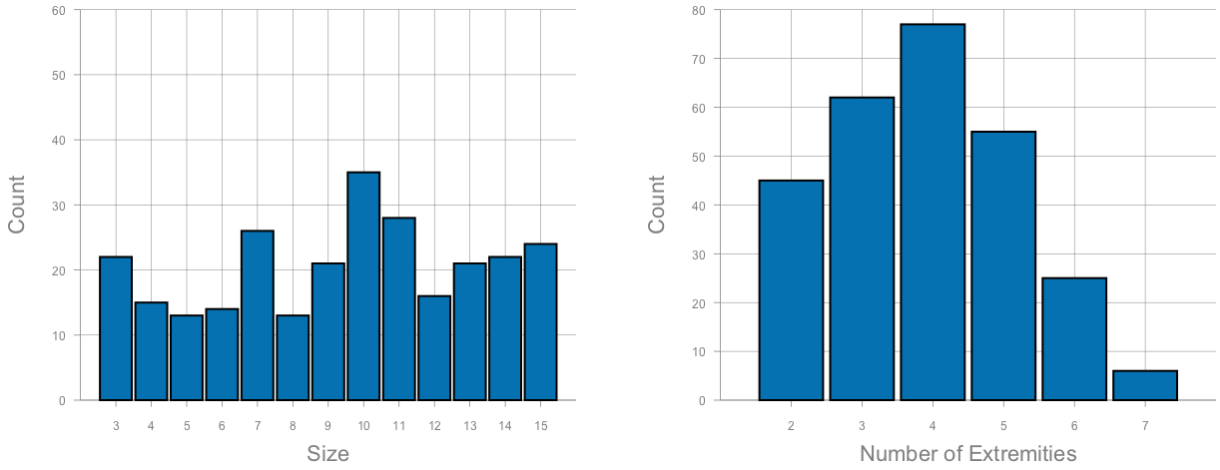
Size The size of a shape is defined by the number of modules. The distribution of sizes is visualised in figure 5a.

Extremities We count the number of extremities, or feet, as a measure of the complexity of an organism. This metric counts the number of modules that are connected on only one side. Figure 5b shows the distribution of the number of shapes per number of extremities. As can be seen 4 extremities is the most common, while shapes with more than 6 extremities are very rare.

Effective Joints The YaMoR modules can be attached to each other either at the moving arm (grey in the model) or on one of the three sides of the module body (green in the model). Attaching a module using its body will leave the arm free, however this arm is not very long, which means it cannot be used to push the organism effectively. In more technical terms, leaving the arm free leads to small moment of force. Attaching a module using the arm effectively lengthens the arm where the force is applied, which increases this moment of force. Having multiple modules in the organism connected with their arms allows a further increase of the moment of force.

We quantify this by counting the number modules that are connected at their arms. These are the *effective joints*. Figure 6a shows the distribution of shapes by number of effective joints. We can see most shapes have between 2 and 7 effective joints, with a few having none at all and very few having more than 8.

[4]In a real implementation of the triangle of life such conflicted shapes should of course be handled automatically



(a) *Distribution of sizes.* **(b)** *Distribution of number of extremities.*

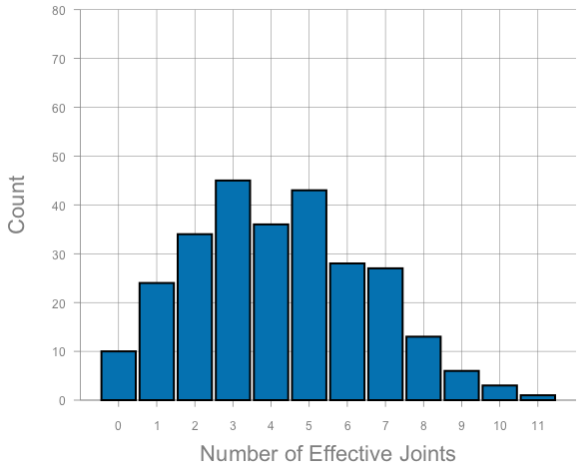
Figure 5: *Distributions of size and nr. of extremities for the 270 randomly generated morphologies.*

Effective Joints per Extremity Finally, we hypothesise that it is not just the number of extremities or the number of effective joints that predict the performance of a shape well, but the ratio of effective joints per extremity. Figure 6b shows the distribution of the effective joints per extremity. We have binned this ratio into intervals of size 0.2, with values less than 0.3 and greater than 2.1 binned in the lowest and highest interval, respectively. An index of 1 effective joint per extremity is by far the most common, but values from 0.5 to 1.7 are also prevalent. A high value indicates a shape that is able to move its body in many ways. With the YaMoR modules, such shapes are usually oblong, or snake-like.

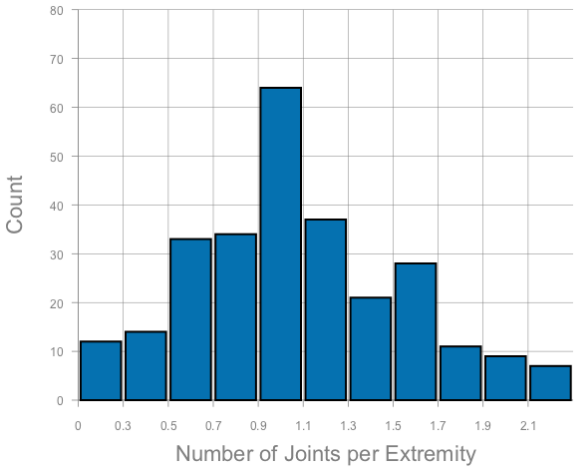
4 Results & Analysis

4.1 Comparing On-line Learners

This section compares the performance of RL PoWER, Simulated Annealing and HyperNEAT as on-line learners of locomotion. These experiments consider only Test Suite I – the designed shapes described in section 3.7. The results for RL PoWER and Simulated Annealing are presented in three groups of plots, one for each morphology (I, T and H shaped) with plots for three sizes (7, 11 and 15 modules). Each plot shows the median performance as well as the interquartile range over 30 replicate runs against time. The performance of the practical optimum (achieved with CMA-ES running off-line with a higher evaluation



(a) *Distribution of number of effective joints.*



(b) *Distribution of number of effective joints per extremity.*

Figure 6: *Distributions of effective joints and nr. of effective joints per extremity for the 270 randomly generated morphologies.*

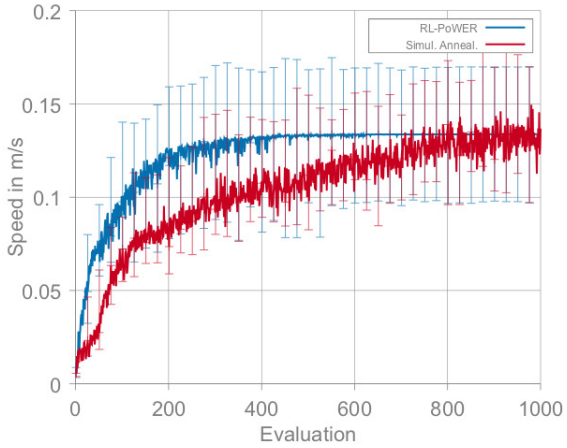
budget) is noted in the relevant tables but not shown in the plots to maintain detail for the comparison of RL PoWER and Simulated Annealing. The HyperNEAT results are discussed separately in section 4.1.2 to highlight the very different on-line dynamics of this algorithm.

The graphs in figures 7, 8 and 9 compare the median speed achieved by RL PoWER and Simulated Annealing in 30 replicate runs. Consider the convergence time of RL PoWER and Simulated Annealing. RL PoWER converges after roughly 400 evaluations (ca 2.6 hours simulated time), regardless of the organisms shape or size. This is a direct consequence of the setting for RL PoWER's variance decay parameter. The convergence time for Simulated Annealing is not as constant and seems to depend on the shape rather than the size. In many cases, particularly for I-shaped organisms, the algorithm is still improving behaviour when the experiments finish.

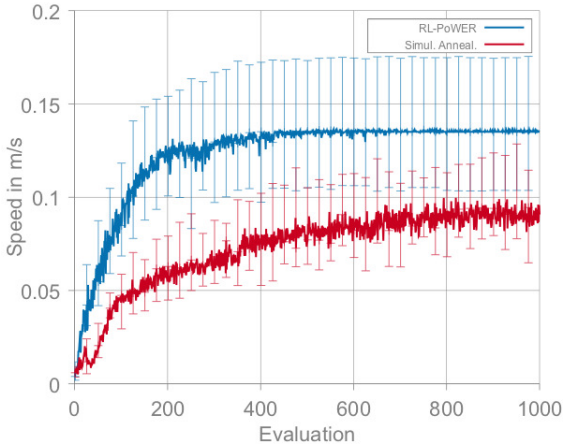
The median speed achieved by the controllers of RL PoWER reaches roughly one third of the practical optimum (the performance reached by CMA-ES running offline). In most cases Simulated Annealing fares worse than RL PoWER, particularly on the T and H shapes. On the I shapes Simulated Annealing fares much better, even

	I	T	H
7	0.9	0.17	**0.016**
11	**15e-04**	0.13	**0.010**
15	**25e-05**	**13e-04**	**30e-04**

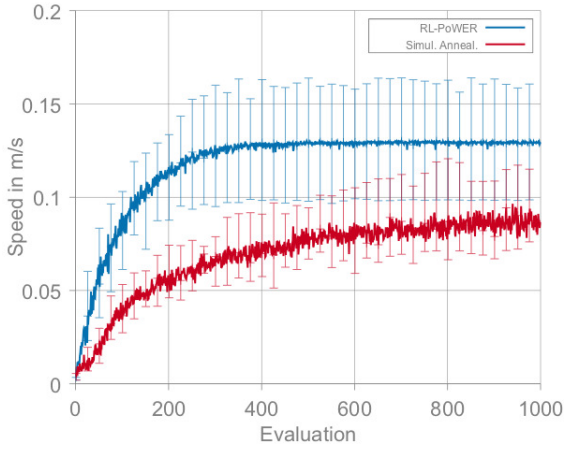
Table 1: *Wilcoxon ranksum test results between the last evaluations of RL PoWER and Simulated Annealing. Significant results ($p < 0.05$) are displayed in bold.*



(a) *Organism I-7*



(b) *Organism I-11*

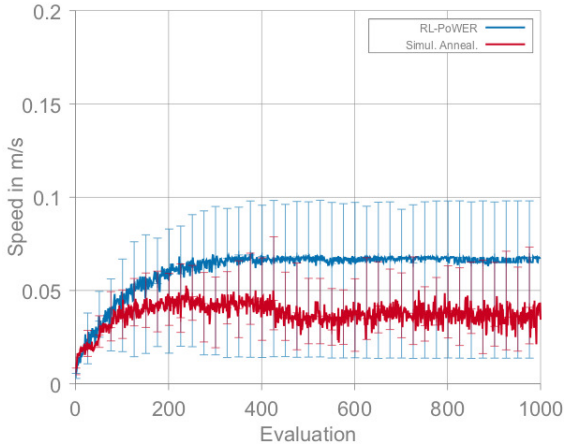


(c) *Organism I-15*

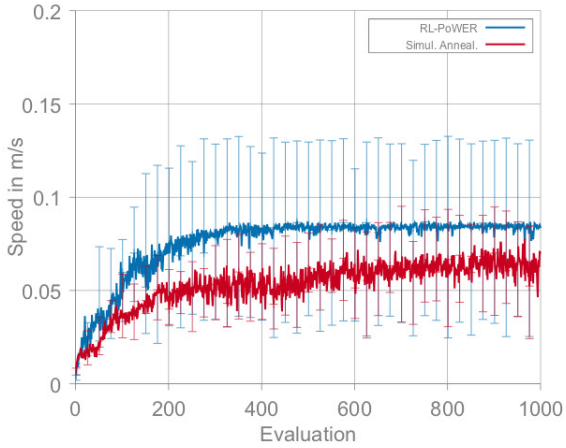
Shape	RL PoWER	Sim. Anneal.	Practical Opti-mum
I-7	0.134m/s	0.137m/s	0.500m/s
I-11	0.134m/s	0.091m/s	0.511m/s
I-15	0.129m/s	0.086m/s	0.383m/s

(d) *Median speed of the last evaluation over 30 runs for RL PoWER and Simulated Annealing compared to the best speed found by CMA-ES*

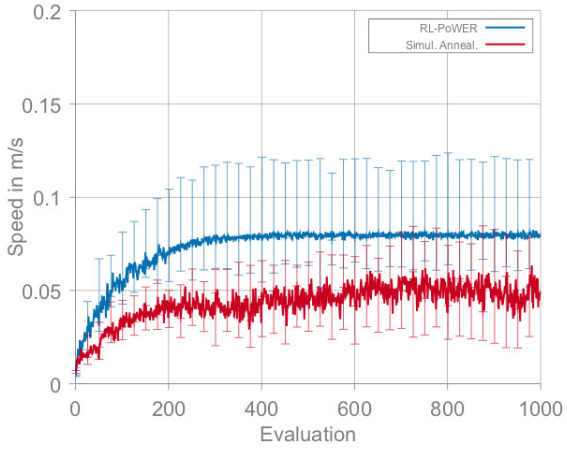
Figure 7: *On-line dynamics of the learning process on the I shapes. The x axis shows time measured by the number of evaluations, the y axis shows evaluation performance (median speed attained, in* m/s*). The blue curve shows the median performance and interquartile range over 30 replicate runs with RL PoWER, the red curve shows the same for Simulated Annealing.*



(a) *Organism T-7*



(b) *Organism T-11*

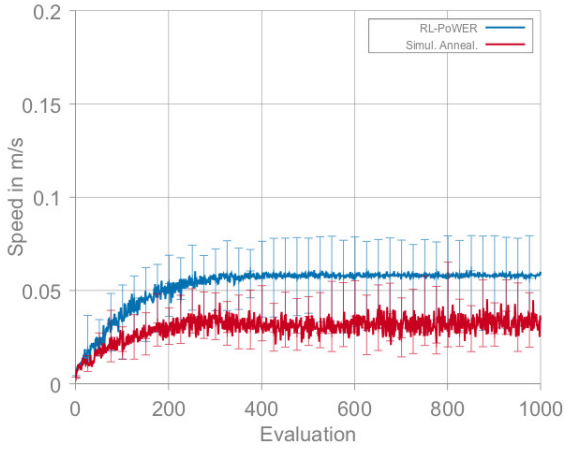


(c) *Organism T-15*

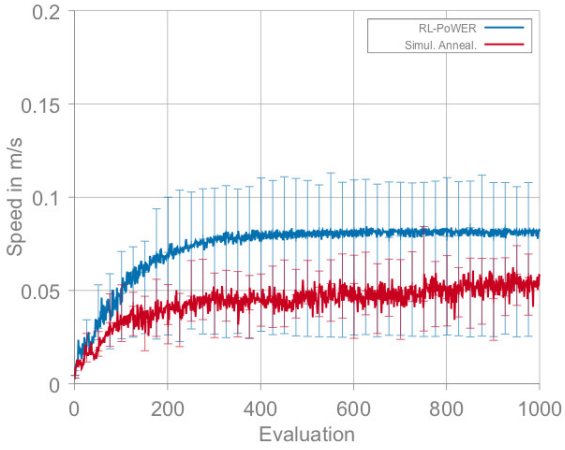
Shape	RL PoWER	Sim. Anneal.	Practical Opti-mum
T-7	0.068m/s	0.034m/s	0.219m/s
T-11	0.086m/s	0.071m/s	0.315m/s
T-15	0.081m/s	0.048m/s	0.293m/s

(d) *Median speed of the last evaluation over 30 runs for RL PoWER and Simulated Annealing compared to the best speed found by CMA-ES*

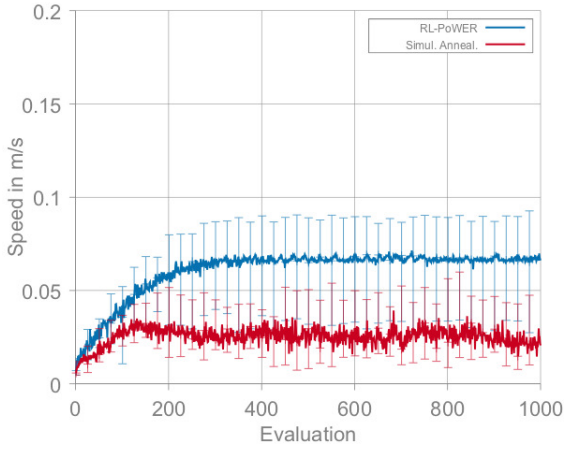
Figure 8: *On-line dynamics of the learning process on the T shapes. The x axis shows time measured by the number of evaluations, the y axis shows evaluation performance (median speed attained, in* m/s*). The blue curve shows the median performance and interquartile range over 30 replicate runs with RL PoWER, the red curve shows the same for Simulated Annealing.*



(a) *Organism H-7*



(b) *Organism H-11*



(c) *Organism H-15*

Shape	RL PoWER	Sim. Anneal.	Practical Optimum
H-7	0.060m/s	0.034m/s	0.241m/s
H-11	0.083m/s	0.057m/s	0.231m/s
H-15	0.066m/s	0.021m/s	0.301m/s

(d) *Median speed of the last evaluation over 30 runs for RL PoWER and Simulated Annealing compared to the best speed found by CMA-ES*

Figure 9: *On-line dynamics of the learning process on the H shapes. The x axis shows time measured by the number of evaluations, the y axis shows evaluation performance (median speed attained, in* m/s*). The blue curve shows the median performance and interquartile range over 30 replicate runs with RL PoWER, the red curve shows the same for Simulated Annealing.*

matching RL PoWER performance for the
I-7 shape. Once converged, RL PoWER provides consistent performance, while Simulated Annealing's performance of consecutive policies is more erratic. To quantify the difference in performance between RL PoWER and Simulated Annealing we performed Wilcoxon ranksum tests to see if their medians differed significantly, the results for these tests can be seen in Table 1. RL PoWER performs significantly better than Simulated Annealing in 6 out of the 9 cases.

A complete run of 1,000 evaluations equates to roughly 6.5 hours of simulated time. In light of current hardware limitations, where reliable operating times of over 4 consecutive hours are rare, this too optimistic a scenario to be feasible in real hardware. Fortunately, RL PoWER converges quite fast and exploratory experiments indicate that we may be able to reduce this without sacrificing performance by using shorter evaluation periods.

Organisms with the same shape but with different size are very similar in performance. Organisms with the same size but different shape, however, show a very large difference in performance. A Kruskal-Wallis test corroborates these results statistically, as shown in table 2. Consider, for instance, the cell for Simulated Annealing and I in table 2a (lower left). The p-value there (10^{-4}) indicates strong evidence that there is a relation between the size of an I-shaped robot and the speed achieved at the end of the 30 replicate runs with Simulated Annealing. The p-value for RL PoWER and 15 in table 2b (upper right) indicates strong evidence of a relation between speed and the kind of shape (I, T or H) for the organisms of size 15.

	I	T	H
RL PoWER	0.792	0.065	0.038
Sim. Anneal.	**e-04**	0.116	**0.002**

(a) *Kruskal-Wallis test results with robots grouped by size.*

	7	11	15
RL PoWER	**e-05**	**e-04**	**e-06**
Sim. Anneal.	**e-10**	**0.0017**	**e-08**

(b) *Kruskal-Wallis test results with robots grouped by shape.*

Table 2: *These tables shows the p-values of a Kruskal-Wallis test comparing the speed of the final controllers in each run. The robots are grouped according to size (left table) or complexity (right table). A low p-value indicates evidence that the size (left) or the shape (right) correlates with organism speed. Significant results ($p < 0.01$) in bold.*

The difference in performance between organisms of the same complexity, but with different sizes is nowhere significant for RL PoWER. For Simulated Annealing different sizes lead to significantly different speeds for the I and the H shapes. With the I shape size 7 being faster than 11 or 15, with the H shape size 11 is fastest. The difference in performance between organisms of the same size, but different shape is significant in all cases. With

either learner the I shape always leads to significantly faster gaits than the T and H shapes, regardless of size. With Simulated Annealing the T-11 shape is also significantly faster than the H-11 shape.

This supports the conclusion that with either algorithm the complexity of the shape has a larger influence on the performance than the size of the shape. The experiments with Test Suite II investigate this further.

4.1.1 Reliability

The interquartile ranges in figures 7 to 9 show substantial variation in the quality achieved in individual runs. To assess the reliability of RL PoWER and Simulated Annealing, figure 10 shows bihistograms with the distribution of runs based on the performance of the last evaluated controller. The blue histograms represent RL PoWER results, the red histograms (extending downwards) represent Simulated Annealing results. In each histogram, a vertical line indicates the corresponding median performance. We define a *bad* run as one that achieves a final speed lower than 0.025 m/s. The graphs show that across the experiments there is a minimum of 1 bad run and a maximum of 9 bad runs for RL PoWER, while there is a minimum of 1 bad run and a maximum of 15 bad runs for Simulated Annealing.

In such bad runs the speed of the organism typically increases until it suddenly drops. The reason for this sudden decrease in speed is that a particular gait during the learning process made the organism lose its balance and flip on its side (I-shaped), head (T-shaped) or back (H-shaped). This poses such a radical change in circumstances that neither learning process was able to find a good controller for the new situation. This falling behaviour can have two causes. First, the on-line paradigm implies that the organism's stance or position is not reset (e.g., to a default stable stance) between evaluations. 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. Second, some organism morphologies may be more prone than others to lose their balance due to detrimental gaits. In [32] such detrimental gaits are filtered out based on knowledge of the organism shape and size, however, we consider a context of arbitrary shapes and sizes, and such filtering is not possible as we do not have prior knowledge of the size or shape.

Although bad runs are undesirable, this need not be a problem in an evolutionary system such as our Triangle of Life scenario, because it calls for a substantial population of robotic organisms. In such a population, bad or unlucky shapes will not be able to reproduce and will disappear over time, but shapes that are well suited for balanced locomotion will be able to prevail.

For both RL PoWER and Simulated Annealing the number of bad runs seems to depend

more on the shape than on the size of the organism. For instance, all I-shaped runs have few bad runs while the T and H shapes lead to substantial numbers of bad runs. Further experiments with randomly generated morphologies in section 4.3 will allow us to assess if this relationship between the number of bad runs and organism complexity holds more generally.

4.1.2 HyperNEAT

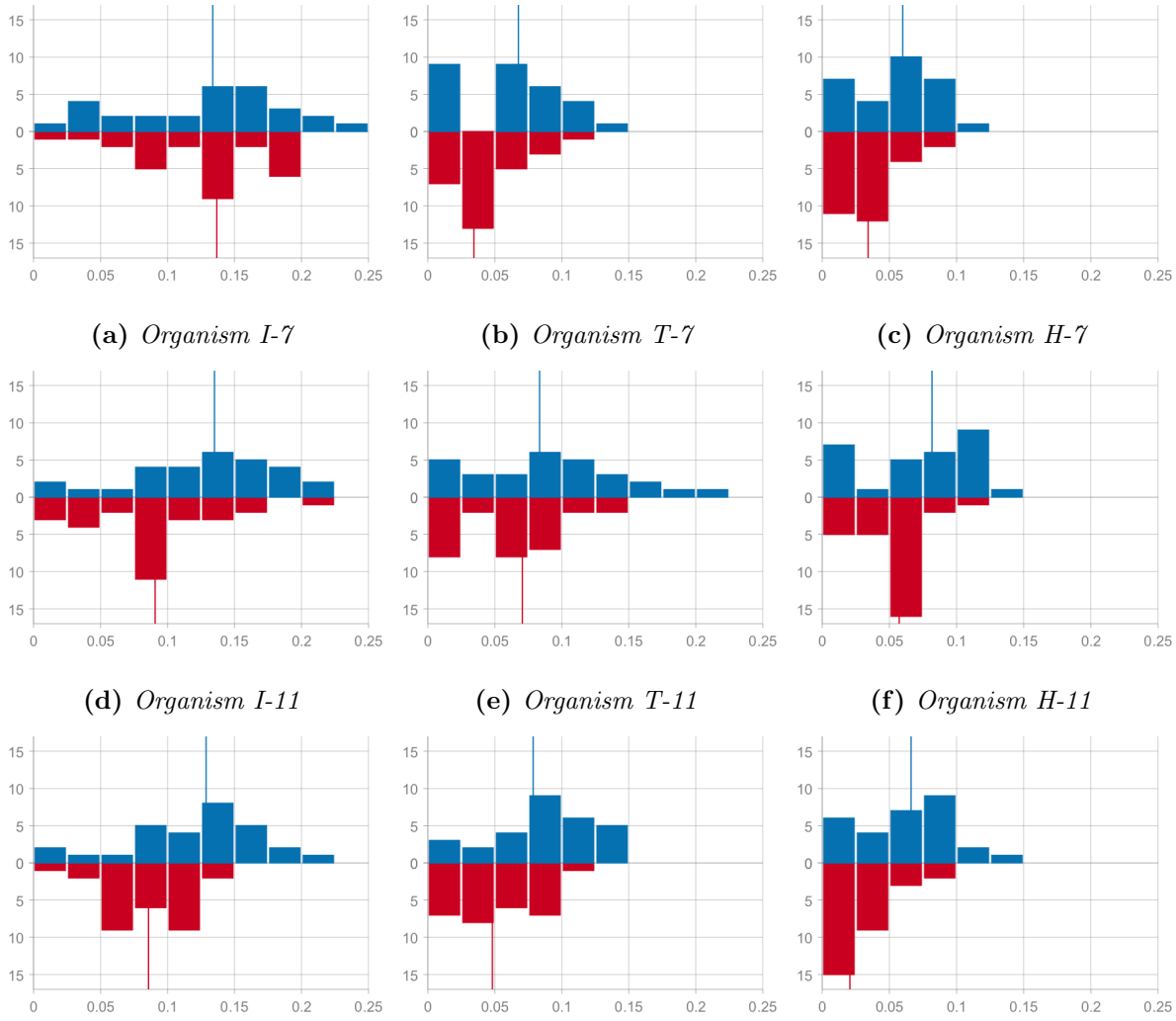
Neural networks are commonly used in evolutionary robotics settings such as these, and as mentioned earlier, HyperNEAT has proved an effective method in this area. Figure 11 shows the results for experiments using HyperNEAT for on-line learning of gaits, similar to the experiments with RL PoWER and Simulated Annealing. The robot runs HyperNEAT locally, serially evaluating individuals in the time-sharing scheme that is typical for encapsulated evolution. The evaluation duration of individual controllers for these experiments is the same as for the RL PoWER and Simulated Annealing experiments, the population size is 25. For further details of the algorithm settings, we refer to the code at `http://github.com/ci-group/tol-project.git`.

Figure 11 shows two sets of results for each of the predefined shapes, accumulated over 30 replicate runs. The blue dots indicate the performance of the individuals of the best run (defined by the last evaluation) and the red dots indicate the median performance over the 30 runs.

The results for HyperNEAT show very competitive controllers, in many instances outperforming the best results of RL-Power and Simulated Annealing. The controllers from the best run (blue dots) are among the best found in our experiments. On the other hand, the median performances are much worse than those for RL-Power: HyperNEAT succeeds in finding very good controllers at the cost of evaluating many poorly performing controllers as well. Therefore, the HyperNEAT results are excluded from Fig. 10.

This highlights an important issue in on-line evolution: because the robot controllers evolve while the robots perform their tasks, the robots' actual performance is determined by the quality of all the controllers they evaluate, not only by the best controllers they consider. In reinforcement learning terms, one must consider the balance between exploration and exploitation when employing evolution in on-line scenarios. This is a radical departure from the optimisation-centred mindset in most evolutionary robotics research that implies off-line development of controllers that do not evolve once deployed.

Thus, algorithms that yield very good results in many off-line neuro-evolution applications are not necessarily useful for on-line learning. In these cases, alternative implementations with a different exploration-exploitation balance must be considered. Silva et al., for

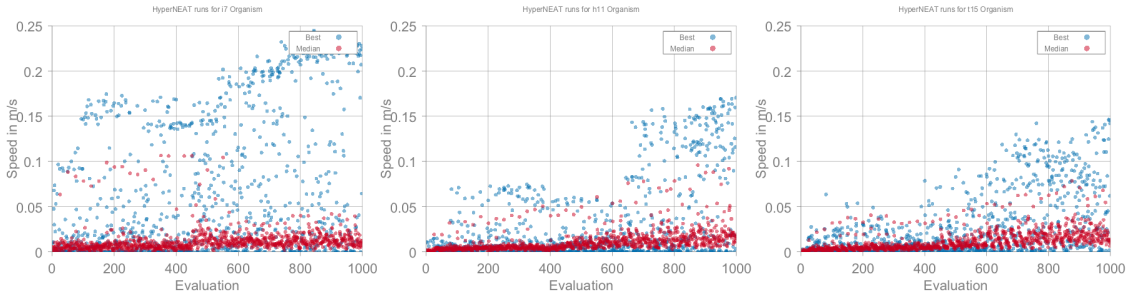


(a) *Organism I-7* **(b)** *Organism T-7* **(c)** *Organism H-7*

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

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

Figure 10: *Reliability of the learning process. The histograms show the number of runs (out of 30) terminating with a speed in a given range. The histogram on the top (in blue) shows RL Power, the histogram on the bottom (in red) shows simulated annealing. The vertical red and blue lines indicate median performance.*



(a) *Organism I-7* **(b)** *Organism H-11* **(c)** *Organism T-15*

Figure 11: *On-line dynamics of the learning process. The x axis represents time (expressed as the number of evaluations) and the y axis represents performance measured by the average speed attained* (m/s)*. The blue marks show the performance of the best run out of 30. The red marks show the median performance of these 30 runs.*

instance, research an on-line variant of NEAT, odNEAT [34], that may provide a basis for a relevant HyperNEAT implementation. Developing such an implementation is beyond the scope of the current comparison, however.

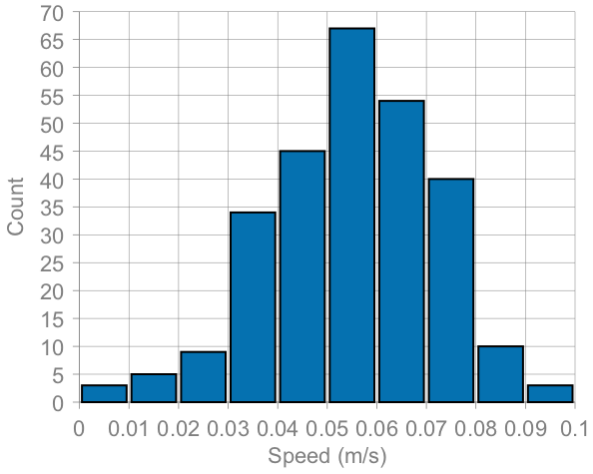
4.2 RL PoWER on Random Shapes

The results on the designed shapes indicate that RL PoWER is a promising candidate learning algorithm in the Triangle of Life context, and the following experiments investigate how this algorithm holds up when applied to randomly generated morphologies. The results show that RL PoWER typically converges within 400 evaluations, so these experiments use 400 evaluations instead of 1,000. The settings for evaluation and recovery time remain the same, at 1,485 and 198 recovery steps (23.76 and 3.168 seconds), respectively. Thus a full run now represents roughly 2.6 hours of simulated time (i.e., the robot 'experiences' 2.6 hours of learning within the simulation), which is within the maximum consecutive running time expected of hardware implementations.

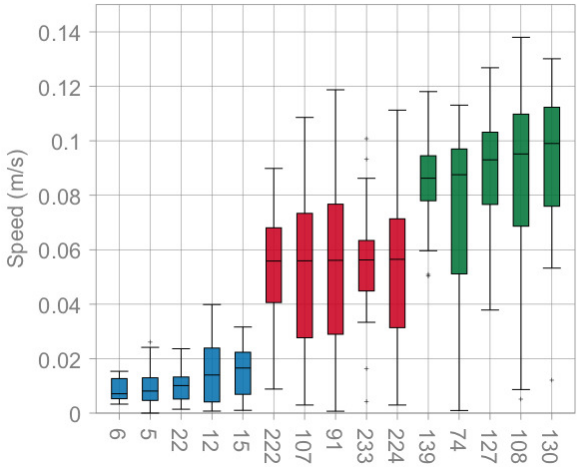
A histogram of the median speed from 30 repetitions for all 270 shapes is displayed in figure 12a. It shows that most shapes reach a speed between 0.03m/s and 0.08m/s. This is somewhat lower than the speeds of the I shapes in the previous section, but is in line with the speeds achieved with the T and H shapes.

A few shapes perform very well with speeds up to almost 0.1m/s, while others lead to a very low median speed. This leads to the conclusion that the generated shapes span a large part of the possible performances.

For a closer look at the performance for particular shapes, 15 shapes are analysed in



(a) *Histogram of median speed for* 30 *replicate runs for all* 270 *shapes.*



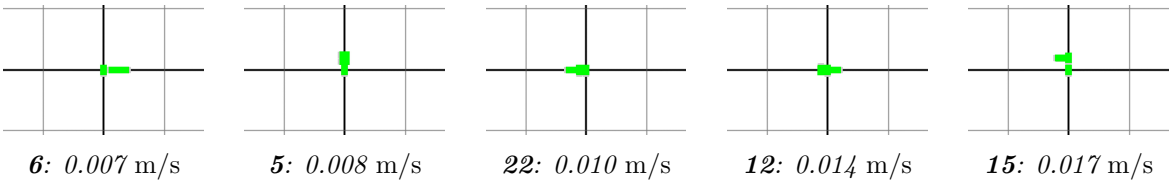
(b) *Speeds achieved in 30 replicate runs with shapes leading to the worst performance, performance around the overall median and best performance. The x-axis is labelled with shape ids.*

Figure 12: *Results of RL PoWER on randomly generated morphologies.*

detail in figure 12b. These are the 5 shapes with the worst final controllers (shown in blue), 5 shapes with around median performance (in red) and the 5 shapes with the best final controllers (shown in green). The plot shows whisker plots with median, quartile, minimum and maximum distance covered with the final controller in the 30 replicate runs. Figures 13 and 14 show simulation screenshots of the shapes of the worst and best shapes respectively.

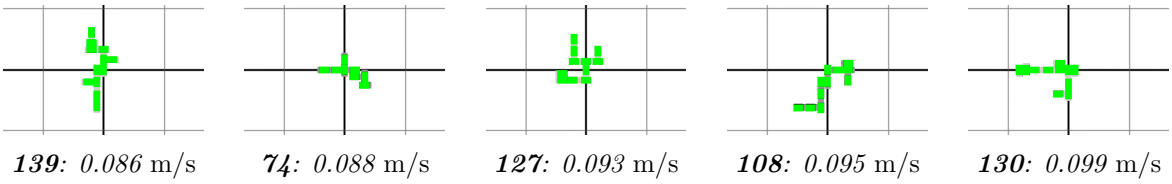
While the 5 worst shapes perform uniformly poorly, the performance spread for faster shapes is comparatively large. For instance, even though the median for shape 74 is among the best 5, its worst performing run is also one of the worst (bottom 10 %). The high spread with individual shapes implies that RL PoWER does sometimes fail to learn a good gait, something that needs to be addressed for a real life implementation of the Triangle of Life. A possible solution could be to introduce a restart mechanism, another would be to tune the evolutionary process in such a way that shapes with low speed still have a reasonable chance to reproduce.

The robots attain speeds ranging from 0 to 0.187m/s, and the median speed across all runs is 0.0562m/s, a performance that is overall somewhat lower than that with the designed shapes. To put this number in perspective: if a similar speed were to be achieved on real hardware (disregarding the reality gap for the purposes of this illustration), a speed of 0.0562m/s would allow an organism to move roughly 600 meters in 3 hours. A real robot



***6**: 0.007* m/s ***5**: 0.008* m/s ***22**: 0.010* m/s ***12**: 0.014* m/s ***15**: 0.017* m/s

Figure 13: *The 5 shapes with the worst median performance with speed indicated.*



***139**: 0.086* m/s ***74**: 0.088* m/s ***127**: 0.093* m/s ***108**: 0.095* m/s ***130**: 0.099* m/s

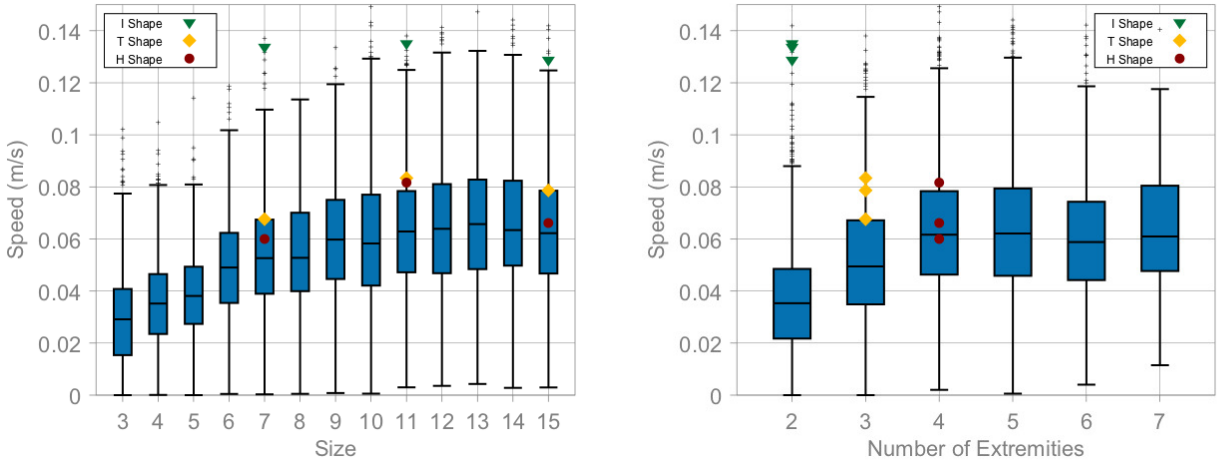
Figure 14: *The 5 shapes that have the best median performance with speed indicated.*

arena will likely be a lot smaller, so these kinds of speeds are more than adequate to travel across the arena and spread an organism's genome to other organisms. This warrants the conclusion that the RL PoWER algorithm works well enough on random shapes to merit further investigation, including trials with hardware implementations.

4.3 Influence of Morphology

The experiments with random shapes can also help initial modelling of the relation between morphology and locomotive success. Figures 13 and 14 show that the worst performing robots are all of size 3, while the best performing shapes are of size 8. Furthermore, the worst shapes have 0 or 1 effective joint, while the best performing shapes have many more. This begs the question how morphological features influence the performance of an organism and which morphological metrics are good predictors for the quality of the learned gait. The following paragraphs analyse the metrics defined in section 3.8 in this respect. For each metric, the speed achieved by the last controller for all repetitions with a particular shape are combined. For reference, the plots in figures 15 and 16 also mark the median speeds achieved with the designed shapes as green triangles (I-shaped), yellow diamonds (T-shaped) and red circles (H-shaped).

Speed vs size Figure 15a shows the speed achieved in the last evaluation grouped by size. There is a clear trend for higher speed with larger size, flattening out for shapes larger than 9. The maximum speed is achieved by a shape with 10 modules, while the lowest speed was by a robot with 3 modules.



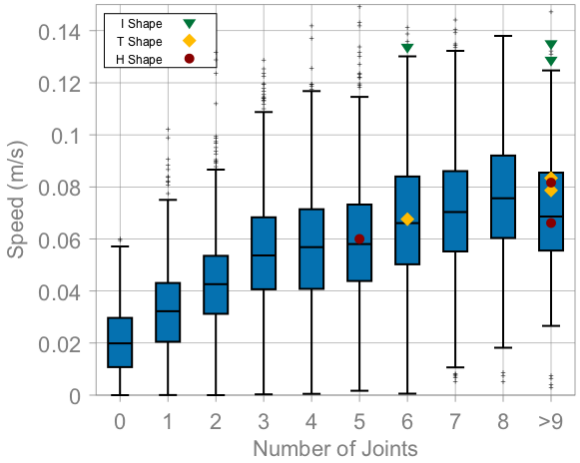
(a) *Speed achieved in the last evaluation for each shape grouped by size. The speeds of the designed shapes is shown as well for reference.*

(b) *Speed achieved in the last evaluation for each shape grouped by number of extremities. The speeds of the designed shapes is shown as well for reference.*

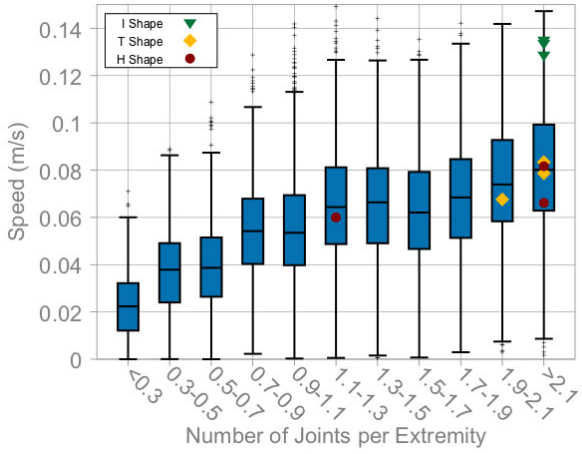
Figure 15: *Speed vs size and vs number of extremities.*

Speed vs number of extremities Figure 15b shows the speed achieved in the final evaluation grouped by the number of extremities. The highest speed is achieved by a shape with 4 extremities, while the worst speed is by a shape with 2 extremities. The trend in this graph is similar to that for size: there is a positive correlation between the number of extremities and speed. Remember that the experiments with the designed shapes show that the I shape, with 2 extremities, was by far the fastest. The T shape, with 3 extremities, was somewhat slower, and the H shape with 4 extremities was the slowest. To answers the question posed at the end of section 4.1.1 whether the positive correlation between number of bad runs and organism complexity holds more generally, we now see that the spread of the speeds in Figure 15 indicates that organisms with any number of extremities can have bad runs.

Speed vs number of effective joints Figure 16a shows the speeds achieved in the final evaluation grouped by the number of effective joints. The results for 10 and 11 effective joints have been grouped with those with 9 effective joints because there were too few shapes with that many effective joints. The best speed was achieved with a shape with 5 effective joints, while the worst speed is achieved with a shape that has 0 effective joints. The plot shows a clear trend of more effective joints leading to higher speeds. Intuitively this makes sense as having articulated limbs allows for complex movement. The definition of effective joint



(a) *Speed achieved in the last evaluation for each shape grouped by number of effective joints. The speeds of the designed shapes is shown as well for reference.*



(b) *Speed achieved in the last evaluation for each shape grouped by the number of joints per extremity. The speeds of the designed shapes is shown as well for reference.*

Figure 16: *Speed vs number of effective joints and vs number of joints per extremity.*

means that only connections that can use the body as a lever and thus create momentum are counted in this metric. The number of effective joints, or some other measure for the ability to apply and leverage force, seems to be a good indicator for the performance of an organism.

Speed vs joints per extremity Figure 16b shows the speeds achieved in the last evaluation grouped by number of joints per extremity. The highest speed is achieved with a shape with more than 2.1 effective joints per extremity, the lowest speed with a shape with less than 0.3 effective joints per extremity.

Again, there is a very clear trend: a higher number of effective joints per extremity implies a morphology that achieves better speed.

4.3.1 Predicting Locomotion Speed

To quantify the relation between the proposed morphological metrics and speed, consider the linear models in table 3. It shows the results of developing linear models for the morphological metrics separately and a model with all features combined (all models significant with a p-value of 0). Of the single features, the number of extremities results in the best fit ($R^2 =$

Nr.	Independent Variables	R^2	F-score	Coefficient	
			$F(1, 8098)$		
1	size	0.16	1540	intercept	0.0286
				size	0.0028
2	extremities	0.246	2650	intercept	0.0319
				extremities	0.0056
3	effective joints	0.0901	802	intercept	0.0318
				effective joints	0.0061
4	effective joints per extremity	0.201	2040	intercept	0.0306
				eff. joints p. extr.	0.0235
			$F(4, 8095)$		
5	size, extremities, effective joints and effective joints per extremity	0.258	705	intercept	0.0178
				size	0.0003
				extremities	0.0011
				joints	0.0030
				eff. joints p. extr.	0.0170

Table 3: *Linear Models trained to predict speed based on various combinations of predictors. In all models the components are significant predictors.*

0.246, i.e., the model explains 24.6% of the variance in speed), however the number of effective joints per extremity offers more discriminatory power as it has the highest coefficient (0.0235, i.e. increasing the number of effective joints per extremity by 1 results in an increase in speed of 0.0235 m/s).

In model 5 all the features are combined, here the R^2 value is 0.258, only a 1% improvement over model 2, but a substantial improvement over model 4. Models with 2 or 3 features were also created, but had R^2 values lower than 0.258 and are omitted here. Model 5 combines a good R^2 with the high coefficient of the number of effective joints per extremity and seems to provide a good starting point for a predictive model for the speed of an organism.

4.4 Comparing Test Suites

In terms of the relation between morphology and speed, the results with the designed shapes seem at odds with those for randomly generated shapes. The speeds achieved with the

designed shapes showed little difference when size increased, but substantial difference when the number of extremities increased. In particular, there was a negative trend with regards to the number of extremities: the H shape with 4 extremities was much slower than the T shape (3 extremities), which in turn was much slower than the I shape (2 extremities).

Consider the median speeds achieved with the designed shapes as indicated in figures 15 and 16.

Clearly, the I shapes have very high median speeds compared to the randomly generated ones. With a median performance around 0.13, the I shapes are at the top end of the graph for every characteristic considered. Also the T shapes perform comparatively well when considering speed vs. size (figure 15a) and speed vs. extremities (figure 15b). However, with the metrics effective joints and effective joints per extremity (figures 16a and 16b) T-shaped robots perform more in line with random shapes. Finally, the H shape has average to low performance across all metrics. In other words, what seemed to be a clear trend in our earlier experiments is likely an anomaly arising from the chosen shapes which disappears under scrutiny of more extensive testing.

Looking at the original shapes in light of the body metrics introduced in section 3.8 we can see that although the size and the number of extremities for the I, T and H shape are in the same range as the generated shapes, the number of effective joints and the number of joints per extremity are very high. So high, in fact, that the numbers are well outside the ranges of the generated shapes.

We can conclude that the set of designed shapes in the first series of experiments is not representative of the set of randomly generated shapes. Which of these two sets is more representative for shapes found during evolutionary exploration of the design space will depend on many factors, for instance the representation and the genetic operators. However in the beginning of the evolutionary process shapes will be more random. Therefore we deem large test suites preferable over small test suites, even if these latter ones are systematically composed.

5 Conclusions

In this paper we addressed the Control Your Own Body problem that arises in populations of modular robotic organisms that evolve in real time. The main problem we considered is that the bodies and controllers of newborn robotic organisms are randomised recombinations of their parents and are unlikely to fit well. Therefore, every 'robot baby' needs to learn to control its own body quickly after birth. In this study we reduced this challenge to an on-line gait learning problem and investigated a solution by applying reinforcement learning. We conducted simulation experiments using robot morphologies of different size and complexity.

Our results show that the RL PoWER algorithm can successfully perform this learning task. It outperforms on-line implementations of Simulated Annealing and HyperNEAT and can find controllers with a speed that is about one third to half of the practical optimum. To put this in perspective, the practical optimum was established by an off-line optimiser whose computational budget was 50 times higher and RL PoWER was used with the default parameter values without tuning it to the problem at hand. RL PoWER converges to a controller after around 400 evaluations (ca 2.6 simulated hours) for all the designed shapes and sizes, making it feasible to learn a gait within the operational time of currently possible hardware implementations.

Successfully learning a gait seems to depend more on the shape than on the size of the robots, while the learning speed of RL PoWER seems independent from either of these factors. The differences between morphologies can be quite substantial: The I shape had very few unlucky 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.

Analysing the relation between morphological metrics and locomotive success of a proposed shape showed that the number of effective joints per extremity is a good predictor by itself. Combining all morphological metrics in a linear model provided a good starting point for a model that can give an a priori indication of an organism's performance. This could benefit the implementation of a robotic ecosystem to enable discarding particularly unpromising proposed morphologies. The morphological metrics used here were not systematically identified; further research into the relation between morphological traits and successful control (including for other tasks than locomotion) could take a more systematic approach that may yield better predictors for locomotive success.

These analyses also showed that that the designed shapes in the first series of experiments are not representative of the shapes that may be found during evolutionary exploration of the design space and that small test suites, even if they are systematically composed, cannot be considered representative without experimental validation.

Further work towards the Triangle of Life is progressing along two lines. First, further experiments with RL PoWER are being conducted with the aim of tuning its parameters in order to reduce the time needed to achieve good gaits. Secondly, efforts are underway to implement and validate this approach on real hardware.

References

[1] E. Aarts and J. Korst. Simulated annealing and boltzmann machines. 1988.

[2] J. Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8):74–85, 2013.

[3] J. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[4] L. Brodbeck, S. Hauser, and F. Iida. Morphological evolution of physical robots through model-free phenotype development. *PLoS One*, 10(6):e0128444, June 2015.

[5] D. J. Christensen, J. C. Larsen, and K. Støy. Fault-tolerant gait learning and morphology optimization of a polymorphic walking robot. *Evolving Systems*, 2013.

[6] D. J. Christensen, U. P. Schultz, and K. Støy. A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots. *Robotics and Autonomous Systems*, 61(9):1021–1035, 2013.

[7] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *IEEE Congress on Evolutionary Computation (CEC) 2009*, pages 2764–2771. IEEE Press, 2009.

[8] M. D'Angelo, B. Weel, and A. Eiben. Online gait learning for modular robots with arbitrary shapes and sizes. In A.-H. Dediu, C. Martín-Vide, B. Truthe, and M. A. Vega-Rodríguez, editors, *Second International Conference on the Theory and Practice of Natural Computing (TPNC 2013)*, number 8273 in LNCS, pages 45–56. Springer, 2013.

[9] S. Doncieux, N. Bredeche, and J.-B. Mouret, editors. *New Horizons in Evolutionary Robotics*, volume 341 of *Studies in Computational Intelligence*. Springer, 2011.

[10] A. Eiben. Grand challenges for evolutionary robotics. *Frontiers in Robotics and AI*, 1(4), 2014.

[11] A. Eiben. In Vivo Veritas: towards the Evolution of Things. In T. Bartz-Beielstein, J. Branke, B. Filipič, and J. Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, volume 8672 of *LNCS*, pages 24–39. Springer, 2014.

[12] A. Eiben and J. Smith. From evolutionary computation to the evolution of things. *Nature*, 521(7553):476–482, May 2015.

[13] A. E. Eiben, N. Bredeche, M. Hoogendoorn, J. Stradner, J. Timmis, A. Tyrrell, and A. Winfield. The triangle of life: Evolving robots in real-time and real-space. In P. Lió, O. Miglino, G. Nicosia, S. Nolfi, and M. Pavone, editors, *Advances in Artificial Life, (ECAL) 2013*, pages 1056–1063. MIT Press, 2013.

[14] A. E. Eiben, S. Kernbach, and E. Haasdijk. Embodied artificial evolution. *Evolutionary Intelligence*, 5(4):261–272, 2012.

[15] D. Floreano, P. Husbands, and S. Nolfi. Evolutionary robotics. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, volume Part G.61, pages 1423–1451. Springer, 2008.

[16] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi. *Gnu Scientific Library: Reference Manual*. Network Theory Ltd., 2009.

[17] J. Gauci and K. Stanley. Generating large-scale neural networks through discovering geometric regularities. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 997–1004, New York, NY, USA, 2007. ACM.

[18] E. Haasdijk, A. E. Eiben, and G. Karafotias. On-line evolution of robot controllers by an encapsulated evolution strategy. In *IEEE Congress on Evolutionary Computation (CEC) 2010*, pages 1–7. IEEE Press, 2010.

[19] E. Haasdijk, A. A. Rusu, and A. E. Eiben. HyperNEAT for locomotion control in modular robots. In G. S. Hornby, L. Sekanina, and P. C. Haddow, editors, *Evolvable Systems: From Biology to Hardware*, volume 5216 of *Lecture Notes in Computer Science*, pages 169–180. Springer, 2010.

[20] H. Hamann, J. Stradner, T. Schmickl, and K. Crailsheim. A hormone-based controller for evolutionary multi-modular robotics: From single modules to gait learning. In *IEEE Congress on Evolutionary Computation (CEC) 2010*, pages 1–8. IEEE Press, 2010.

[21] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[22] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653, 2008.

[23] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in artificial life*, pages 704–720. Springer, 1995.

[24] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji. Automatic locomotion design and experiments for a modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 10(3):314–325, 2005.

[25] A. Kamimura, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata. 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*, volume 3, pages 2370–2377. IEEE Press, 2004.

[26] J. Kober and J. Peters. Learning motor primitives for robotics. In *IEEE International Conference on Robotics and Automation (ICRA) 2009*, pages 2112–2118. IEEE Press, 2009.

[27] R. Möckel, C. Jaquier, K. Drapel, E. Dittrich, A. Upegui, and A. Ijspeert. YaMoR and Bluemove – an autonomous modular robot with Bluetooth interface for exploring adaptive locomotion. In M. O. Tokhi, G. Virk, and M. A. Hossain, editors, *Proceedings of the 8th International Conference on Climbing and Walking Robots (CLAWAR) 2005*, pages 685–692. Springer, 2006.

[28] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, 2000.

[29] G. Parker. Co-evolving model parameters for anytime learning in evolutionary robotics. *Robotics and Autonomous Systems*, 33:13–30, 2000.

[30] G. Parker. Evolving gaits for hexapod robots using cyclic genetic algorithms. *International Journal of General Systems*, 34(3):301–315, 2005.

[31] C. Rossi and A. Eiben. Simultaneous versus incremental learning of multiple skills by modular robots. *Evolutionary Intelligence*, 7(2):119–131, 2014.

[32] H. Shen, J. Yosinski, P. Kormushev, D. G. Caldwell, and H. Lipson. Learning fast quadruped robot gaits with the RL PoWER spline parameterization. *Cybernetics and Information Technologies*, 12(3):66–75, 2012.

[33] W.-M. Shen, B. Salemi, and P. Will. Hormones for self-reconfigurable robots. In E. Pagello, F. Groen, T. Arai, R. Dillman, and A. Stentz, editors, *Proceedings of the 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 918–925. IOS Press, 2000.

[34] F. Silva, P. Urbano, S. Oliveira, and A. L. Christensen. odneat: An algorithm for distributed online, onboard evolution of robot behaviours. In *Artificial Life*, volume 13, pages 251–258, 2012.

[35] A. Spröwitz, R. Moeckel, J. Maye, and A. J. Ijspeert. 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.

[36] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

[37] V. Trianni. *Evolutionary Swarm Robotics – Evolving Self-Organising Behaviours in Groups of Autonomous Robots*, volume 108 of *Studies in Computational Intelligence*. Springer, 2008.

[38] P. Vargas, E. D. Paolo, I. Harvey, and P. Husbands, editors. *The Horizons of Evolutionary Robotics*. MIT Press, 2014.

[39] L. Wang, K. Tan, and C. Chew. *Evolutionary Robotics: from Algorithms to Implementations*, volume 28 of *World Scientific Series in Robotics and Intelligent Systems*. World Scientific, 2006.

[40] B. Weel, E. Crosato, J. Heinerman, E. Haasdijk, and A. Eiben. A robotic ecosystem with evolvable minds and bodies. In *Proceedings of the 2014 IEEE International Conference on Evolvable Systems (ICES)*, pages 165–172. IEEE, 2014.

[41] M. Yim. A reconfigurable modular robot with many modes of locomotion. In *Proceedings of International Conference on Advanced Mechatronics*, pages 283–288. Japan Society of Mechanical Engineers, Tokio, Japan, 1993.

[42] J. Yosinski, J. Clune, D. Hidalgo, S. Nguyen, J. Zagal, and H. Lipson. Evolving robot gaits in hardware: the HyperNEAT generative encoding vs. parameter optimization. In T. Lenaerts, M. Giacobini, H. Bersini, P. Bourgine, M. Dorigo, and R. Doursat, editors, *Advances in Artificial Life, (ECAL) 2011*, pages 890–897. MIT Press, 2011.