Published in Evolvable Systems: From Biology to Hardware 2010*; Gianluca Tempesti, Andy Tyrrell and Julian Miller, editors. The original publication is available at* `www.springerlink.com`.

HyperNEAT for Locomotion Control in Modular Robots

Evert Haasdijk, Andrei A. Rusu, and A.E. Eiben

Dept. of Computer Science, Vrije Universiteit Amsterdam, The Netherlands
e.haasdijk@few.vu.nl, andrei.alex.rusu@gmail.com, gusz@few.vu.nl
http://www.cs.vu.nl/ci/

Abstract. In an application where autonomous robots can amalgamate spontaneously into arbitrary organisms, the individual robots cannot know a priori at which location in an organism they will end up. If the organism is to be controlled autonomously by the constituent robots, an evolutionary algorithm that evolves the controllers can only develop a single genome that will have to suffice for every individual robot. However, the robots should show different behaviour depending on their position in an organism, meaning their phenotype should be different depending on their location. In this paper, we demonstrate a solution for this problem using the HyperNEAT generative encoding technique with differentiated genome expression. We develop controllers for organism locomotion with obstacle avoidance as a proof of concept. Finally, we identify promising directions for further research.

1 Introduction

The research presented in this paper was undertaken as part of the European research project SYMBRION: Symbiotic Evolutionary Robot Organisms.[1] As the name suggests, a key objective of the project is the evolution of robot organisms – structures consisting of physically connected individual robots like those in Fig. 1 for tasks that an unconnected group of individual robots cannot cope with.

In SYMBRION, individual robots are fully autonomous and viable as individuals, while they have the ability to dock with each other and so aggregate into organisms, becoming modules (cells) within the organism. Once in organism mode, the modules share energy and control, acting autonomously but in co-ordination. Co-ordination is inherently distributed, without central control. Such emergent organisms are not made to last forever: they can separate to become a swarm of individual robots once more. The individual robots are then available for the formation of new, possibly differently shaped, organisms. This high level of flexibility implies challenging requirements for robot controllers. Firstly, an individual robot needs a controller that works appropriately within differently shaped organisms. For instance, the robot should be able to act within a "snake", a twenty-legged body, or a "dog" with four legs, a head and a tail. Furthermore, any robot should be able to function at different positions of any given organism shape, e.g., at the head as well as in the middle of a snake. As an example of a task for an organism that requires co-ordinated control of the robots/modules, consider locomotion; obviously a key ability for the organism to perform meaningful tasks. In this paper

[1] EU grant agreement 216342.

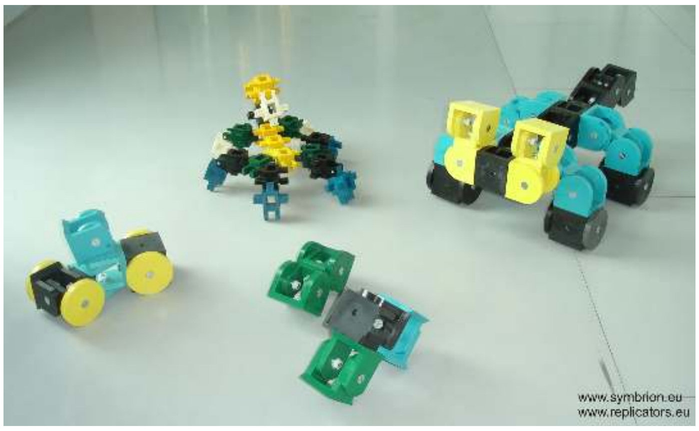


Fig. 1. Illustration of possible SYMBRION organisms.

we leave shared energy for what it is and address the challenge of robot controllers that work appropriately at different positions of a given organism shape. In particular, we seek an evolutionary method that can produce such controllers for arbitrary organism shapes (note: not a single controller for arbitrary shapes but a single developmental technique).

Let us first clarify the difference between a robot controller and the evolvable code that represents that controller. In general, a robot controller is a structurally and procedurally complex entity that directly determines the robots behaviour. When using evolutionary methods for controller design, controllers are seen as phenotypes that are represented by (structurally simpler) pieces of code, called genotypes. The phenotypes are then perceived as expressions of the genetic code in the genotype through a possibly complex mapping. The fitness of an individual (typically: task performance) is then determined by the phenotype. Meanwhile, –conforming to biological principles of evolution– it is only the genotypes that undergo evolutionary operators (mutation and/or crossover), not the phenotypes. Distinguishing in this way between phenotype (the actual controller) and the genotype that encodes it allows us to rephrase the challenge: we seek an evolutionary method that is capable of generating genotypes that give rise to controllers that work appropriately at different positions of a given organism.

Because it is unknown a priori at which location in an organism a particular robot will end up, the robots must have a single genotype that encodes appropriate controllers for each location: the group of robots is literally homogeneous. However, they should have the flexibility to show different behaviour depending on their position in an organism. This means that their phenotype should be different, depending on their location. For instance, a module that forms part of a quadruped's backbone has a different role and thus requires a different controller than does a module that makes up, say, a hip joint (in biological terms, the expression of the genotypes must be influenced by the environment).

We argue that an evolutionary algorithm with a generative encoding presents a natural way to meet these requirements. As noted by D'Ambrosio and Stanley in [3], variation on a policy theme distributed across space is reminiscent of the regular spatial patterns for which generative encodings are known [8, 11]. For our purposes, generative encodings offer the benefit that the genome can be interpreted multiple times with variations. In our own bodies, this is exactly what happens when our DNA is expressed: for instance, variations in expression cause each of the segments of our spines to be similar yet specifically differentiated for their role within the spine as a whole. Enabling similar differentiation when expressing the genome as controllers for the organism's modules allows the development of varying, specialised functionality. For this procedure of varying the expression of a genotype to create specialised controllers for the organism's modules, we coin the phrase *modular differentiation*.[2]

Of course, specialisation can also be achieved by separately evolving specialist controllers for (collections of) joints, vertebrae, etc. and selecting the appropriate controller as needed. While such divide-and-conquer tactics have resulted in successful locomotion, the underlying decomposition is inherently specific to a particular morphology and must be performed manually. Also, it runs the risk of introducing constraints and biases that limit the quality of solutions cf. [7, 2, and citations therein].

Locomotion of an organism that consists of autonomous modules can be viewed as a task of a co-operating team of individual agents, with each module constituting an autonomous agent. Although extending the scope of their findings to this scenario might be tenuous, Waibel et al. have shown that for tasks requiring co-operation, homogeneous teams outperform heterogeneous ones [12]. The modular differentiation approach allows us to enjoy the best of both worlds: it exploits the benefit that homogeneous teams enjoy without sacrificing the advantages of specialisation.

The individual robots that make up the organism also have their own sensory capabilities that allow them, for instance, to steer the organism away from obstacles they detect. Consequently, we seek controllers that put sensor information –specifically, obstacle detection– to use: they should implement *reactive* control in addition to habitual motion patterns such as found in [4, 6].

Summarising, the aim of this paper is to present an evolutionary algorithm that combines generative encoding and modular differentiation to evolve reactive, co-ordinated, autonomous modular controllers for organism locomotion.

2 Generative Encoding Description

The generative encoding we use is called HyperNEAT [9], which evolves artificial neural networks with the principles of the widely used NeuroEvolution of Augmented Topologies (NEAT) algorithm [10]. HyperNEAT evolves a particular type of artificial neural network, called a Compositional Pattern Producing Network (CPPN). While traditionally, artificial neural networks typically contain only sigmoid functions, CPPNs can employ a mixture of many other functions.

[2] The term *modular differentiation* was chosen as an analogy to developmental biology's cellular differentiation, the process by which a less specialised cell becomes a more specialised cell type.

A CPPN defines a function that can be employed, for instance, to assign grayscale values to pixels in an image, as was done to generate the picture in Fig. 2[3], which highlights important attributes of the CPPNs that evolve in HyperNEAT: they tend to produce designs with a large degree of regularity, symmetry and repetition. Often, patterns are repeated with slight variations and at varying scales. The consequent layout can be perceived as modular with variations.

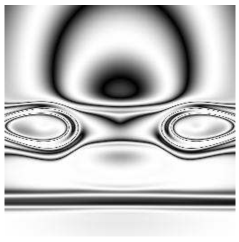


Fig. 2. A CPPN-generated grayscale image.

HyperNEAT uses the CPPNs as an indirect encoding, so the CPPNs do not constitute the controllers for the robot modules themselves. Instead, the CPPNs are used to set up the artificial neural networks that do control the robots. To avoid confusion, these artificial neural nets that form the phenotype are usually referred to as *substrates*. To define a substrate, the CPPN specifies the weight for every possible connection in the template substrate; the connection weight between two nodes is determined by querying the CPPN with the two nodes' co-ordinates, which then returns the required connection weight. Often, the distance between the nodes is passed into the CPPN as well. This method of generating the substrate assigns meaning to the location of the neural net's nodes, implying that HyperNEAT has the unique ability to exploit the geometry of a problem [9]: if the geometric disposition of the nodes in the substrate represents relevant information, HyperNEAT can use that information.

HyperNEAT has been successfully used in many applications, maybe most pertinently to develop gaits for four-legged robots by Clune et al. [2]. There, Clune et al. used HyperNEAT to develop monolithic, central controllers for a table-shaped robot. This robot did not, in contrast to the organisms considered here, consist of multiple modules, so modular differentiation could not play a role in controller development. Moreover, no obstacle detection was employed and therefore control could not avoid obstacles as we aim to do.

3 Experimental Set-up

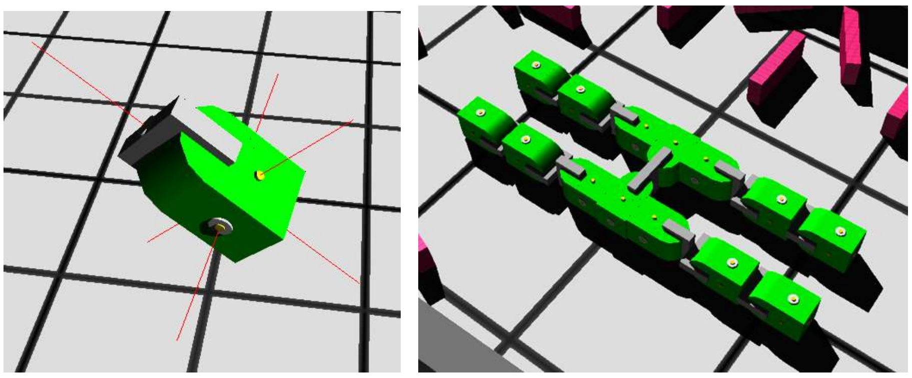
We evolved controllers for locomotion of a quadruped organism consisting of 14 simple modules as a proof of concept. Experiments were conducted in the well-known *Webots*[4] simulation platform.

3.1 Modules and Organism

We based the modules on the YaMoR [6] oscillators. These consist of a solid body and an oscillating arm, offering one degree of freedom. We added two extra connectors, bringing the total to 4, which are situated as follows: one on the joint's arm, one on the opposite side of the module, and two on the left and right of the mobile joint, in the motion plane. See Fig. 3(a) for a rendition of a module. For obstacle detection, we

[3] See `http://picbreeder.org/` for more examples and information.
[4] `http://www.cyberbotics.com/`

added 6 distance sensors with very limited range, indicated in figure 3(a) by the thin lines emerging from the module. They are distributed on all sides of the module.



(a) Basic module (b) Quadruped organism

Fig. 3. Module and organism.

The module positions in the quadruped organism shown in Fig. 3(b) are inspired by joint disposition in natural insect legs. The central modules allow for mid-body flexibility. The organism is completely symmetrical around its center point.

In these experiments, each module's controller has the sole task of setting the target angle value for the actuator in each control step to achieve locomotion for the organism. The individual modules that make up our organism are simpler than those being developed in the SYMBRION project [5] but for the purposes of organism locomotion have a similar degrees of freedom and sensors.

3.2 Control

Each module within the organism operates autonomously and with only local interaction. As described in Sec. 2, each module is controlled by its own neural network, or substrate, controller. The nodes of the substrate are arranged in three layers of nine nodes each as shown in Fig. 4. Links between nodes run only in one direction and only between consecutive layers. The nodes have sigmoid activation functions.

Inputs consist of processed sensory information: when a new object appears in the range of the sensors, a 'new presence' flag in the centre of the input layer (labelled 'self' in Fig. 4), is set to -1. To compute the occurrence of a new object, the distance sensors are queried in each control step and the returned values are compared to the values in the previous step. If at least one sensor gives a reading increase above 50% of the maximum activation level, this is interpreted as the detection of a new object in the perceptual range of the module, and the center input layer node is activated (with values -1).

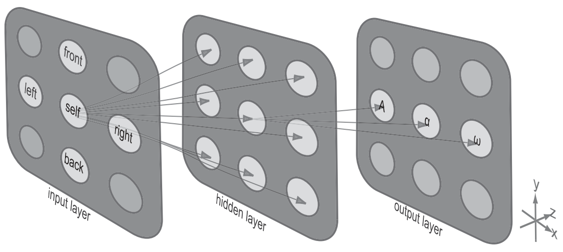


Fig. 4. The substrate layout for the locomotion task. Connections are shown as illustration; actual connectivity is determined by the CPPN.

This scheme is loosely inspired by the biological processing of olfactory information, which triggers strong responses primarily at the initiation of new stimuli, but then develops adaptation ('fatigue').[13] Note that the continued presence or removal of an object from a sensor's range is not signalled.

Up to four adjacent controllers (of any directly connected modules) send their own flag: these values are set in the substrate input corresponding to the geometric position of connectors. In the 3x3 input layer the central node accounts for the current module, and nodes above and below, to the left and to the right of that node account for the modules connected using the front, back, left and right connectors, respectively.

This very primitive, distributed, object detection scheme is intended to allow for simple but effective reactions to obstacles.

If no perceptual changes are detected by the sensors of the current module, or the modules connected to it, the substrate inputs are 0, allowing for default non-reactive locomotive behaviour as specified by the output layer biases. Note, that this default behaviour actually requires no interaction with other modules at all and the organism moves by virtue of the modules acting in splendid isolation.

Producing a successful gait with such a reactive framework is harder than a non-reactive one (which is actually implemented by the output layer's biases), because the modules are subjected to potentially different "perceptual histories" at every evaluation. However, this scheme exposes the changes in behaviour to the evolutionary algorithm and allows for adjustments to the base angle, speed and amplitude as responses to perceived objects.

The output layer provides three values for the computation of the target angle of the joint in each control step: α (reference angle), $\mathcal{A}$ (deviation amplitude from the reference angle) and ω (angular speed of the oscillation). The target angle is computed as follows:

$$\alpha_{target} = \alpha + \mathcal{A} \cdot sin(\pi\omega t + id) \tag{1}$$

with t the current time-step and id a number between 1 and 14 which identifies the current module within the organism with no geometric meaning. This encoding of the joint's motion allows for both static and dynamic joints, with specific oscillation amplitudes and speeds. The modules are out of phase by a number of steps determined by their position in the organism. This is important for generating some motion in the initial stages of the evolutionary process. This encoding scheme was devised for its effective task decomposition into concepts of speed, amplitude and a base angle.

3.3 Modular Differentiation

To achieve modular differentiation, we extend the information passed to the CPPN when determining the connection weight between two nodes in the substrate. Remember that normally the connection weight is determined by querying the CPPN with the two nodes' co-ordinates, often passing the distance between the nodes into the CPPN as well. In addition, we pass the CPPN inputs locating the module for which we are generating connection weights within the organism. By virtue of these extra inputs, each module in the organism will have a different set of connection weights in its neural net controller, but the underlying phenotype (i.e., the CPPN) is the same throughout the organism.

To be precise, the substrate weights are determined by querying the CPPN with the corresponding co-ordinates for the source and destination nodes in 3 dimensions $\langle x, y, z\rangle$ and the relative position of the module in the organism on a two dimensional plane $\langle t_1, t_2\rangle$, illustrated in Fig. 5.

We also use four delta inputs: Δx, Δy and Δz are the respective co-ordinate value differences, while Δt is the Euclidean distance to the centre of the organism shape.

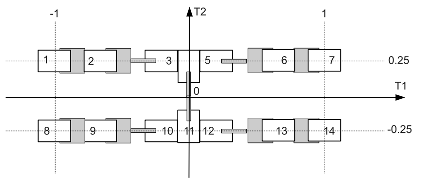


Fig. 5. Distribution of modules in the $\langle t_1, t_2\rangle$ coordinate space. Modules are labelled with their *id*s.

As an example, consider the link between two nodes at co-ordinates $\langle 1, 0, 1\rangle$ and $\langle 0, 1, 0\rangle$ in the substrate. To determine the weight for that connection the CPPN would be queried with nine values that pertain to the two nodes themselves: the six original co-ordinate values and three Δ-values that denote the differences for the x, y and z co-ordinates ($\Delta x = 1, \Delta y = 1, \Delta z = 1$). Additionally, we pass three values to differentiate between modules: for module 6 in Fig. 5, for example, we pass $t_1 = 0.66, t_2 = 0.25$ and $\Delta t = \sqrt{0.66^2 + 0.25^2}$, while for module 11 these values are $t_1 = 0, t_2 = -0.25$ and $\Delta t = \sqrt{0 + 0.25^2}$.

Links for which the CPPN returns values below 90% are ignored, so the CPPN's output is interpreted as a link's relevance measure, and only very strong stimulatory and inhibitory links are kept. The 90% threshold was established empirically. The perceptual scheme introduces a lot of noise directly into the values that determine the motion patterns, so only very strong links are worth keeping.

3.4 Task and Evolution

We ran a series of simulations in the arena depicted in Fig. 6. The task for the organism was to move the whole body along the corridor of which the walls are too high to scale. The corridor is littered with bricks. The organism starts roughly in middle of the corridor. Bricks and walls are detected when they are in the (short) range of each module's distance sensors. Bricks can be moved, but walls cannot. This allows for a "perceptual" difference between them, since bricks are more dynamic and will typically

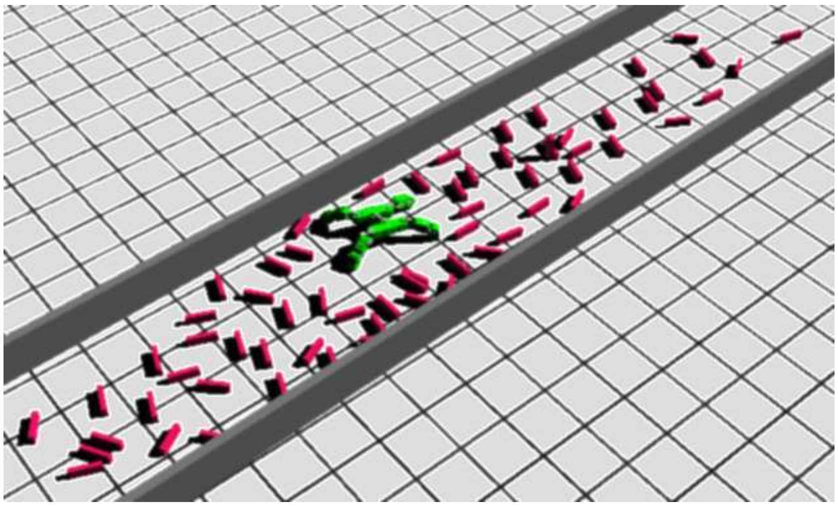


Fig. 6. Experimental setting: a corridor with bricks and walls.

activate sensors which walls will not, e.g. underneath the body. The organism needs to adjust its gait to steer away from walls, but not be deterred by mere stacks of bricks.

Each evaluation lasts 20 simulated seconds for a total of around 80 control steps. Each CPPN is evaluated 3 times on the same task, to get a better approximation of its fitness. Fitness increases exponentially with the final distance from origin achieved by the organism, and the average height of the middle section, it is computed as follows:

$$f(CPPN) = e^{(d_{origin} * 0.95^{(\frac{d_{travelled}}{d_{origin}} - 1)} + h_{avg})} \tag{2}$$

with d_{origin} the distance from the origin after 20 seconds, $d_{travelled}$ the total distance travelled and h_{avg} the average height from the floor of the body's centre during the 20 second evaluation period. The $\frac{d_{travelled}}{d_{origin}} - 1$ part measures the effectiveness of the overall gait: the final distance from origin is scaled down to penalise ineffective gaits that do not move in one consistent direction. The inclusion of h_{avg} promotes individuals that can raise their bodies. As the distance-related part of the fitness formula quickly becomes an order of magnitude larger than the average height, its effects are felt mainly in the initial stages of evolution. We used Jason Gauci's publicly available C++ implementation of HyperNEAT, version 2.6.[5] Apart from a population size of 10, we used the settings as found in that implementation's TicTacToe experiment. We did not engage in further tuning of parameters or thorough analysis of alternative fitness calculations since the experiments provide a proof of concept rather than a comprehensive analysis.

4 Results and Analysis

The evolved gaits we observed were smooth and seemed natural with the organism moving in a controlled, co-ordinated manner using cyclical motion patterns. In the later stages of evolution, motion patterns often exhibit left-right symmetry, replacing the initial phase difference to produce useful gaits. They gave the impression that the organism would happily walk for hours on end without faltering as the organism returned to a neatly poised stance after every step.

The sensory input was often seen to be used with the organism lifting a leg higher than normal to avoid a brick, as illustrated in Fig. 7. Because the bricks can also be shoved aside, this kind of behaviour did not always emerge, but that it does at all is a clear indication that reactive controllers do evolve in this set-up.

Figure 8 shows the development of fitness over 25 repeats of the experiment. The centre line shows the median of the best of every generation over 25 runs, with the bars extending from the lower to the upper quartile. Considering the exponential nature of Eq. 2, the median fitness after 150 generations of circa 15 equates to more than 2.5 metres travelled. The lower quartile after 150 generations, at 10, equates to travelling ca. 2.3 metres. For values of 20 or higher, the organism actually reaches the end of the corridor after 3 metres.

To analyse the effect of modular differentiation in the organism and the reactivity of the controllers we analyse the substrate outputs of a high fitness individual. The networks use sigmoid functions with outputs between -1 and 1, which are afterwards linearly rescaled to the full ranges of the effectors. For simplicity we omit the scaling here, and show raw network output values. Figure 9 shows substrate outputs for the three output nodes (see substrate in figure 4) for all 14 controllers over 80 control steps. The horizontal base lines indicate the substrate output in unexcited state, i.e., when no obstacles are detected. If the controllers were identical, these lines would obviously overlap for all modules: the different levels we see are the result of modular differentiation. Jags in the plots indicate reactions to perceptual input (detecting an obstacle), either direct or via a neighbouring module. Note that not all modules react with the same intensity or at the same time, further proof of modular differentiation.

Figure 9(a) shows the outputs for the base angle α; many of the outputs remain constant throughout the experiment: controllers that do not use sensory inputs to set the base angles. The number of lines we can distinguish indicate that modular differentia-

[5] `http://eplex.cs.ucf.edu/software.html#gaucij_HyperNEAT`

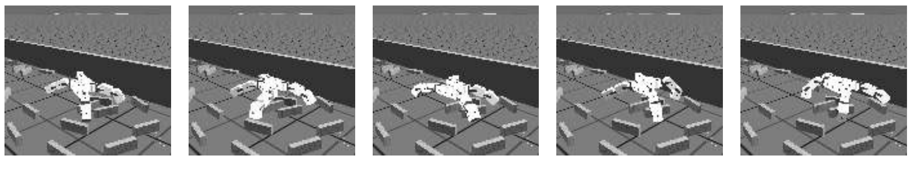


Fig. 7. Locomotion while negotiating an obstacle.

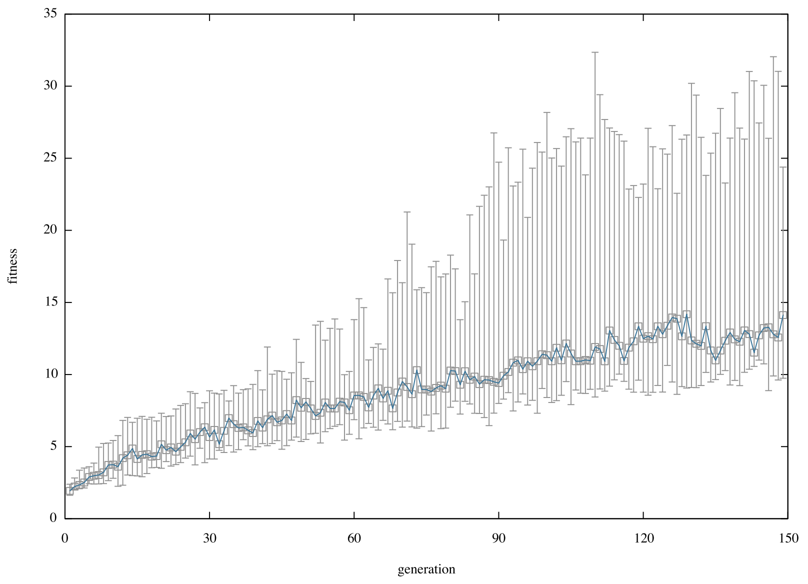


Fig. 8. Fitness plot over 150 generations. The centre line shows the median of each generation's best individual over 25 repeats, the bars extend from the lower to upper quartile.

tion leads to some specialisation. The variation of the three non-constant plots results from obstacle detection, but the magnitude of the changes is small.

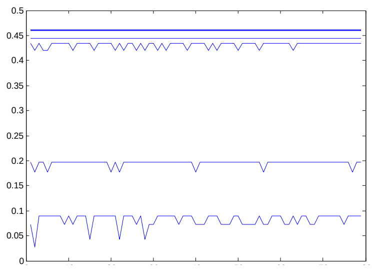
Figure 9(b) shows that the outputs for angular speed (ω nodes) are almost equal for all modules (note the scale). Moreover, no perceptual information is used, since the outputs are constant. This parameter barely differentiates modules.

By far the most diverse behaviour is shown in Fig. 9(c), which depicts the amplitude node outputs. All controllers use perceptual information to set amplitude values, and the magnitude of the changes is as big as 0.3 in absolute difference, in some cases. Also, there is a high degree of specialisation, since the default output levels range from -0.6 to 0.4.

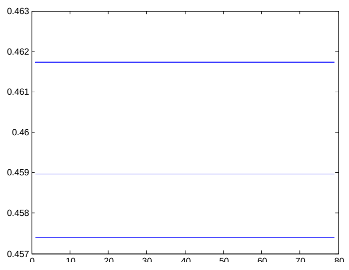
5 Conclusion and Future Work

Using HyperNEAT's generative encoding technique and modular differentiation, we have designed an evolutionary algorithm to develop homogeneous yet specialised controllers for modules within a multi-robot organism. We showed that this algorithm can successfully develop a reactive quadruped gait. The individual robots' controllers act autonomously and with only local exchange of information but in a co-ordanated manner to allow successful locomotion of a given organism.

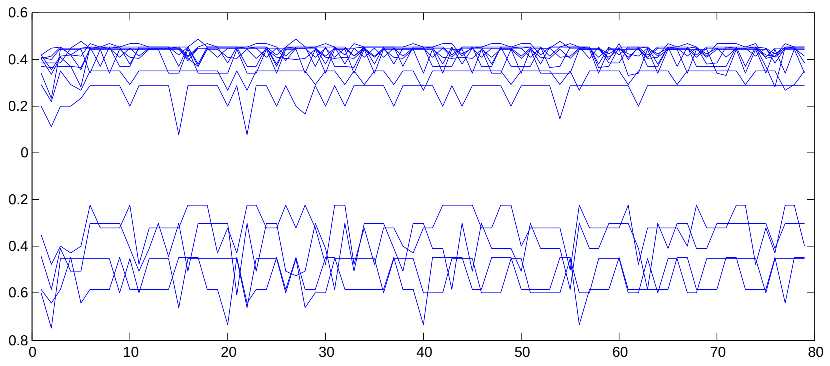
Analysis of the substrate output of all modules over the course of an evaluation showed considerable differences in activation between modules, indicating adaptation



(a) Base angle (α)



(b) Angular speed (ω)



(c) Amplitude $(\mathcal{A})$

Fig. 9. Substrate outputs for all modules of a high-fitness organism over 80 time-steps.

of module controllers to their particular position in the organism as the result of modular differentiation.

The controllers incorporate sensory feedback from the modules' obstacle sensors, resulting in the CPPN encoding multiple motion patterns. The base pattern is determined by the substrate output layer biases (used when no obstacles are detected and the remaining controller network is not activated). The CPPN also encodes the changes to this default behaviour, different for each perceptual flag combination, which directly activates the network. Instead of exchanging information about the motion pattern, the modules send information about detected obstacles to any directly connected neighbours.This way perceptual information propagates locally and progressively, as the new object also enters the sensory range of adjacent modules.

Analysis showed that the primitive "perceptual flag" sensory scheme can successfully switch policies for all modules, for this particular individual the most notable changes affecting amplitude values.

Further study of the perceptual scheme described here is required to asses its effectiveness in arenas of different shapes and scales. A promising avenue of further research leads towards an implementation of the HyperNEAT modular differentiation approach for on-line adaptation of controllers for emergent rather than pre-defined organism morphologies. Future research will also combine the use of CPPNs to generate organism morphology as well as controllers for the constituent modules.

Acknowledgements This work was made possible by the European Union FET Proactive Intiative: Pervasive Adaptation funding the SYMBRION project under grant agreement 216342. The authors would like to thank Nicolas Bredèche and other partners in the SYMBRION consortium, Jeff Clune and Selmar Smit for many inspirational discussions on the topics presented here. Jason Gauci, Ken Stanley and other members of the very active HyperNEAT community have provided invaluable help.

References

1. *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, Trondheim, May 18-21 2009. IEEE Press.
2. Jeff Clune, Benjamin E. Beckmann, Charles Ofria, and Robert T. Pennock. Evolving coordinated quadruped gaits with the hyperneat generative encoding. In CEC-2009 [1].
3. David B. D'Ambrosio and Kenneth O. Stanley. Generative encoding for multiagent learning. In Conor Ryan and Maarten Keijzer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2008)*. ACM, 2008.
4. Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642 – 653, 2008. Robotics and Neuroscience.
5. S. Kernbach, E. Meister, O. Scholz, R. Humza, J. Liedke, L. Ricotti, J. Jemai, J. Havlik, and W. Liu. Evolutionary robotics: The next-generation-platform for on-line and on-board artificial evolution. In CEC-2009 [1], pages 1079 –1086.
6. Daniel Marbach and Auke Jan Ijspeert. Online optimization of modular robot locomotion. In *Proceedings of the IEEE Int. Conference on Mechatronics and Automation (ICMA 2005)*, pages 248–253. IEEE Press, 2005.
7. Stefano Nolfi and Dario Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, 2000.
8. Kenneth O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, June 2007. Special issue on developmental systems.
9. Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.
10. Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, March 2002.
11. Kenneth O. Stanley and Risto Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.
12. Markus Waibel, Laurent Keller, and Dario Floreano. Genetic Team Composition and Level of Selection in the Evolution of Cooperation. *IEEE Transactions on Evolutionary Computation*, 13(3):648–660, 2009.
13. Philip Whitfield and D.M. Stoddard. *Hearing, Taste, and Smell; Pathways of Perception*. Torstar Books, Inc., New York, NY, 1984.