An Improved System for Artificial Creatures Evolution

Thomas Miconi[1] and Alastair Channon[1]
[1]University of Birmingham, Edgbaston B152TT, Birmingham, UK
t.miconi@cs.bham.ac.uk

Abstract

We present our complete reimplementation of Karl Sims' system for evolving and coevolving autonomous creatures in a physically realistic three-dimensional (3D) environment. Creatures are articulated structures composed of rigid blocks and controlled by embedded neural networks. The main differences with Sims are, first, the use of standard McCulloch-Pitts neurons (instead of a set of ad hoc, complex functional neurons) and, second, an improved genetic encoding and developmental system (allowing fine-grained control of neural connections in duplicated morphological features, and replication-exaptation processes).

This paper expands upon a previous version of our system (Miconi and Channon, 2005) which implemented a subset of features present in Sims' system, and dealt with simple evolutionary experiments based on external fitness functions only: the present paper extends the feature set proposed by Sims, and describes the results of experiments based on the 'box-grabbing' coevolutionary task introduced by Sims. We provide a detailed description of our model and freely accessible source code. We describe some of our results, including an analysis of evolved neural controllers. To the best of our knowledge, our work is the first replication of Sims' efforts to achieve results comparable to Sims' in efficiency and complexity, with standard neurons and realistic Newtonian physics.

Introduction

This paper describes a platform for the evolution of autonomous articulated structures ("creatures") in a physically realistic 3D environment. This platform is by and large based on the model introduced by Sims (Sims, 1994b; Sims, 1994a), with modifications.

While evolving artificial creatures can be an interesting and useful endeavour in itself (e.g. in relation to robotics, especially modular robotics (Mesot, 2004)), our primary motivation in implementing such a platform is to create a superior experimental tool for the study of artificial evolution and, most importantly, coevolution. Physical embodiment, free morphologies and arbitrary behaviours open a practically limitless range of possibilities for evolution to explore for any particular problem. This richness is particularly important in coevolution, where the fitness landscape is determined, at least in part, by the features of the evolving individuals themselves. In this context, the flexibility of artificial creatures provides a fertile ground for complex adaptations and counter-adaptations, with the additional benefit that these adaptations are often intuitively interpretable by visual inspection.

We are therefore interested in building a very specific type of system: a physically realistic 3D environment in which articulated creature can evolve towards potentially arbitrary levels of complexity[1]. We add the requirement that this system must be powerful enough to allow for reliable, consistent success: the evolutionary system must be flexible enough to allow competent behaviours to emerge routinely. Finally, we want to do this with a very general architecture, without any ad hoc machinery to favour a specific type of behaviour. To the best of our knowledge, the closest thing to such a system so far has been Karl Sims' model of evolving creatures.

Since the publication of Sims' seminal papers, several authors have described virtual creatures systems. However these systems differed from Sims' in several aspects, often by simplifying the physics drastically (e.g. the Framsticks system (Komosinski, 2000)), or by concentrating their attention towards a specific aspect of creature evolution (with less emphasis on evolutionary performance), or both. For example, (Bongard and Pfeifer, 2001) constructed and studied a nature-inspired developmental system based on genetic regulatory networks. (Ray, 2000) evolved creatures interactively for aesthetic merit. (Hornby and Pollack, 2001) introduced a grammar-based generative system for 'stick-figure' creatures with simplified physics. Direct replications of Sims' work have been scarce and usually incomplete (e.g. (Taylor and Massey, 2001))[2].

[1]Note that we are *not* asserting that evolution will mechanically impose an increase in complexity.

[2]A more detailed review of the field can be found in (Miconi and Channon, 2005)

The system

In this article we describe our own model for the evolution of artificial creatures in a physically realistic 3D environment. This model is broadly similar to Sims', but with important differences. Our work brings three contributions with respect to Sims':

1. Our creatures are controlled by standard neural networks, based on classical McCulloch & Pitts neurons with sigmoid or radial activation functions. This is in contrast with the ad hoc functional neurons used by Sims. While Sims' approach was entirely justified given the seminal aspect of his work, we believe that using standard neurons provides a higher level of generality to our model: creatures cannot rely on complex neurons to generate behaviours, they must build these behaviours 'from scratch' (including simple, vital behaviours such as oscillations).

2. We introduce extensions to the genetic-developmental system described by Sims. First, we address a problem not mentioned by Sims: structures which are replicated by the developmental system, either through symmetry (reflection) or through recursion (segmentation), initially possess identical neural information and thus cannot be independently controlled or provide distinct information to ancestor limbs. We solve this problem by adding genetic flags which control the actual wiring. Second, we make it possible for developmental duplications of genetic nodes to be transcribed back into the genome, creating several (initially similar) genetic nodes which may then evolve independently, in analogy with the duplication-exaptation process found in Nature.

3. We provide a complete description of our system, as well as the original source code. The lack of information on crucial aspects of Sims' system has been an obstacle to replication. The program described in this paper is freely available under the terms of the GNU General Public License (GPL) at `http://www.cs.bham.ac.uk/~txm/creatures/` together with video samples.

An early version of our platform was described in a previous paper (Miconi and Channon, 2005). This early version implemented a subset of the features described by Sims: in particular, recursive replication of limbs (which allows for segmentation of body plans) was not possible. Furthermore only simple experiments based on external, hand-defined fitness functions were reported. The present paper extends the feature set proposed by Sims, and describes the results of experiments based on the 'box-grabbing' coevolutionary task introduced by Sims.

Creature morphology

In the following sections we provide a broad description of our system, stressing both similarities and differences with Sims' model. In order to facilitate comparisons, our description deliberately follows the same organisation as Sims (Sims, 1994b), section by section. Note that many of the basic features described in these sections can also be found in our previous paper (Miconi and Channon, 2005).

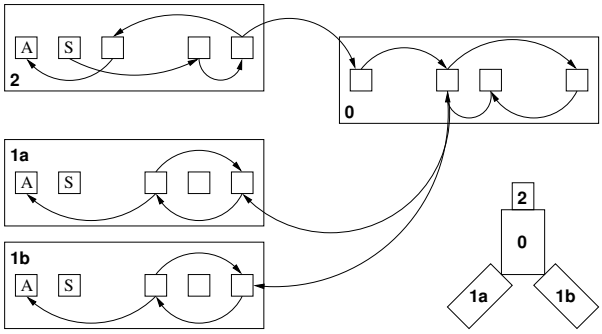

Figure 1: Organisation of a fictional creature pictured in the bottom-right corner. Limb 0 has no sensor (S) or actuator (A). Limb 1 is reflected into two symmetric limbs 1a and 1b, which share the same morphologic and neural information.

As in Sims' model, the creatures are branching structures composed of rigid 3D blocks. The blocks (or "limbs") are connected to their parent limb by a hinge joint - except for the first, "root" limb which has no parent. The genetic specification of a creature is given as a graph of nodes. Each of these nodes contain morphologic and neural information about one limb. Each node is responsible for storing the description of its limb's physical connection with its parent node's limb, removing the need for connections to carry their own information, as is the case in Sims' model.

The morphologic information in each genetic node specifies the *dimensions* of the limb (i.e. width, length and height), the *orientation* of this limb with regard to its parent (in the form of two parameters indicating polar angles with the *xz* and the *xy* planes, i.e. longitude and latitude, in the frame of reference of the parent limb; these two parameters are discrete multiples of $\pi/8$), the *axial direction* of the hinge joint which may be either horizontal or vertical (i.e. aligned either with the *y* or with the *z* axis of the limb), and a boolean flag for *reflection* which governs symmetric replication along the *xz* plane of its parent (see section on Genome Expression). A limb also contains *neural information*, as described in the next section.

Creature control and neural organisation

Our creatures are controlled by neural networks. As in Sims' model, each limb contains a set of neurons. Genetic information about a given neuron specifies the *activation function* for this neuron, a threshold/bias parameter θ taken in the $[-1,1]$ range, and connection information. The activation function may be either a sigmoid ($\frac{1}{1+\exp^{-(\sigma+\theta)}}$) or the hyperbolic tangent $\tanh(\sigma+\theta)$ where σ is the weighted sum

of inputs (the difference between sigmoid and tanh is that the first has values in $[0,1]$ while the latter has values in $[-1,1]$). Connection information specifies, for each connection, the source of this connection (i.e. the neuron whose output is received through this connection) and a weight in the $[-1,1]$ range. Neurons can only be connected with other neurons from adjacent limbs, or from the root limb. Each neuron may receive a variable number of connections, up to a maximum value (3 in the present experiments).

The most important difference with Sims' model lies in the choice of standard neurons with traditional activation functions, in contrast to Sims' large set of functions (including arithmetic operations and oscillators). An important consequence of this simpler set of functions is that there is no trivial way for evolution to generate oscillators or other cyclic forms of behaviour, which are necessary for any sustained locomotion to take place. Such behaviours have to emerge out of the interaction between several neurons, assembled together under the guidance of evolution. A more practical consequence is that in our model, each neuron may have an arbitrary number of inputs (up to a maximum value), by contrast to Sims' neurons which had a fixed number of inputs, dependent on their function.

Sensors and actuators

Sensor neurons and actuator neurons are handled specially. In these experiments two types of sensors are used: proprioceptors and external sensors. Proprioceptive neurons measure the current angle formed by the hinge joint to which this neuron's limb is attached, scaled within the $[-1,1]$ range. External sensors come in two types, which measure the *x*-distance of the limb containing the sensor to either the trunk of an opponent, or to an inert box. An *x*-distance is the distance between the centre of mass of the limb and the centre of mass of the object, along the *x*-axis of the frame of reference of the limb in which the sensor exists (i.e. the *x* component, in the frame of reference of this limb, of the vector joining the centre of this limb to that of the detected object). The outputs of external sensors are squashed through a *tanh* function. Actuator neurons command the movement of limbs, that is, the desired angular velocity around their joint. Their inputs are defined similarly as other neurons, but their activation function is always a scaled hyperbolic tangent of the form $MaxSpeed * \tanh(\sigma)$, where *MaxSpeed* is a system constant. In our model, sensor neurons do not receive any connection from any other neuron, and no neuron may receive a connection from an actuator neuron. Sensors and actuators are, respectively, pure sources and pure sinks of data.

A difference with Sims' model is that an actuator does not specify a force or a torque, but a *desired angular velocity*. The physics simulator implements a motor at each joint, which will constantly attempt to reach the desired speed, with the constraint that the total torque it exerts cannot be larger than a specified maximum. This maximum is a system constant. This mechanism corresponds to a very simple model of servomotors.

Each limb has exactly one actuator and one proprioceptor. It may have other neurons, including external sensors, within a maximum number (in the current experiments the maximum number of neurons for each limb in addition to the actuator and the proprioceptor is 2). Note that while each limb has a sensor and an actuator, there is no requirement that they should receive or send connections from or to other neurons: connections are established in a random manner and no connection toward these special neurons is explicitly enforced. Thus each limb is free to use its sensor and actuator, or not, depending on how its network evolves. This is equivalent to Sims' model.

Expression of the Genome: the Developmental System

When a creature is to be generated from its genotype, a simple developmental system translates the genotype into a corresponding phenotype, and may introduce additional complexity if the genetic information dictates it. Our system uses developmental features similar to those introduced by Sims (bilateral symmetry and segmental replication). To fully exploit these features, we introduce control flags which enforce fine-grained control of the neural connections in replicated limbs. We also introduce a new mutation operator, recursion unrolling, which allows developmental replications to be transcribed back into the genome.

Reflection

Symmetry in our model is implemented somewhat differently than in Sims'. In our model, each genetic node (corresponding to a limb) may possess a "reflection" flag, which means that when this node is read and the corresponding limb attached to its parent, a symmetric copy of this limb will also be created. Any further sub-limbs will similarly be duplicated in a symmetric fashion, which leads to the appearance of bilaterally symmetric branches. Our present design allows for only one type of symmetry, namely symmetry along the parent's *xz* plane. When a given limb is randomly generated, its reflection flag is set with probability P_{ref} (for this paper, $P_{ref} = 0.25$).

Symmetric replication introduces information flow issues. When a limb is duplicated by reflection, all genetic information is duplicated in the process, including neural information. A consequence of this duplication is that *a given limb cannot distinguish information it sends to, or receives from, either of its symmetric sub-limbs*. Because neurons from both symmetric sub-limbs share the same connection information, they will receive identical connections (and information) from the same neurons in the parent. Similarly, any connection that the sub-limbs send to the parent will point to the same neuron in the parent, and information from

both sub-limbs will be merged at that point. Thus, although both limbs may behave in different manners due to their separate inner neural networks (which may react independently to different sensor information), they will not be able to send distinct information to the parent, or to receive distinct information from it. Sims does not mention this problem, or document his solution to it, in his papers.

We address this problem in the following way: every connection has a special $Reftype$ flag, which can take one of three values: Original, Symmetric, or Both. When a connection (as specified by the genome) originates from a neuron that exists in a reflected limb, then the actual connection in the resulting creature will be connected either to the original version of the limb, or to its symmetric copy, or to both, depending on the value of its $Reftype$ flag. If $Reftype$ is 'Both', then this connection will carry the average of the outputs of the two neurons.

Similarly, if neurons from the two instances of a reflected limb carry a connection originating from their common ancestral limb, the $Reftype$ flag is used to determine the actual wiring, that is, whether only the original instance, or the symmetric copy, or both, will receive input from the parent limb.

Segmentation

In Sims' model, a loop in the genetic graph corresponds to a set of limbs which is repeated a certain number of times. Connections between limbs specify a recursive limit, which is the maximum number of times this connection should be followed when in a recursive cycle. Some connections may also be marked as "terminal", meaning that they will only be applied when the recursive limit is reached, thus allowing for specific "tailing" structures at the end of repeated sequences. This, in essence, is a simple and effective model of *segmentation*, that is, the repetition of homologous modules arranged sequentially, as apparent in many animals (vertebrates, arthropods, anellidae, etc.).

We import this feature in our model, with the restriction that only self-loops are allowed: a loop can only exist between a node and itself. No other loops within the graph can exist. This allows for bio-inspired segmentation (repetition of similar segments), while preventing the appearance of extravagant body plans (such as, say, a human body in which the thumb would contain a "loop" to the thorax).

Segmentation brings in the same information flow issues as were discussed above about symmetry. Imagine that a certain node has a recursive loop to itself, inducing its replication into similar segments. How can we allow for communication between segments ? In the genome, the information about a connection specifies the genetic node (and the neuron within this node) from which this connection originates. But when a connection refers to the same node as the one within which it exists, and the node is recursively replicated, we must decide which instance of the node is actually referred to (so that connections could occur within the same limb, between one limb and its recursive predecessor, or between one limb and its recursive successor).

We address this issue in the same manner as with symmetry. Each connection also carries a *Rectype* flag, commanding its behaviour under recursion, which can take any of three values: Dad, Son, or Same. If a connection for a given neuron originates from the same genetic node as that in which the neuron exists, and this neuron is recursively duplicated, then the value of *Rectype* determine the actual wiring of this connection: the value 'Dad' indicates that this connection should go from one instance to its predecessor in the recursion (so obviously it will not exist in the first, 'original' instance); the value 'Son' indicates that this connection should go from one instance to its successor in the recursion (so it will not exist in the last, 'terminal' instance). The value 'Same' indicates that this connection should be understood to originate from the same instance and will therefore be present in all instances.

Recursion Unrolling

A common source of novelty in Nature is the duplication-exaptation process: one part is duplicated in two, originally identical elements, then the features of these elements diverge and assume different roles. The versatility of arthropod appendages is a striking example of these mechanisms. Our system allows for similar duplication-exaptation patterns indirectly, through a mutation operator called recursion unrolling: a recursive cycle (one genetic node which specifies its own recursive replication during development) is *unrolled* in the genome, that is, it is developed as it would be in a final body, and the resulting limbs are transcribed back into as many new, independent genetic nodes. Originally these new nodes are almost identical (except for neural connections within themselves and between each other, which depend on the flags described in the previous section), but from now on they can evolve independently. This method may be in opposition with the central dogma of genetics (information flows from the genotype to the phenotype), but is not Lamarckian: it is a macro-mutation which occurs randomly, without using selective information, thus respecting the Darwinian mechanism of 'blind' mutations. This mechanism is not present in Sims' system.

Creature evolution

Genetic operators

We use three genetic operators, similar to those used by Sims, plus one addition.

Crossover is performed by simply aligning the genetic nodes of both parents in two rows, then building a new list of genetic nodes by concatenating the left part of one parent with the right part of the other.

Grafting corresponds to the removal of a branch (i.e. a limb and all its sub-limbs), and its replacement by a branch

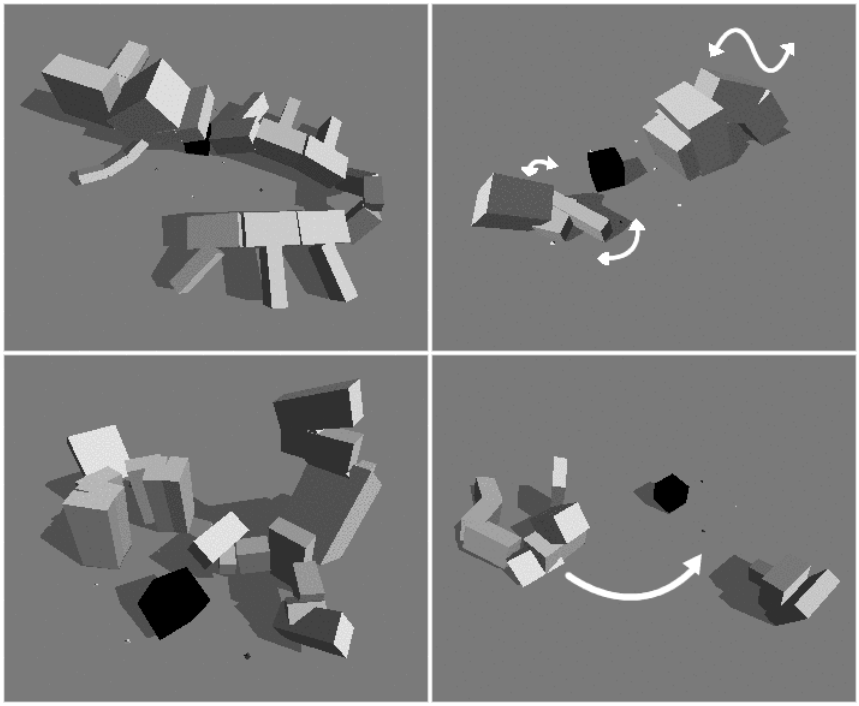

Figure 2: Creatures from four different runs. In the first run, a two-armed 'centipede' creature tries to wrestle the box from a massive creature which uses its mass to pin it down (the centipede uses both segmental and symmetric duplication of limbs). In the second picture the left creature uses horizontal wiggling to crawl forward, while the right creature uses ample vertical sinusoidal movement from its 'tail' to propel itself against the ground. In the third picture one creature has just knocked the ball away, using the constant rotatory movement from its right 'arm' (the left arm is fixed, a clear example of differential neural control over two duplicated limbs); but the other creature, using its accordion-like segmented appendages, will eventually displace it and interpose itself between its opponent and the cube. In the fourth picture, the creature uses a sinusoidal crawling movement from its tail to move past the cube, and relies on its sensor to turn around it, thereby cutting off its opponent.

taken from another individual. Connectivity information is adapted and maintained: the neurons of the trunk establish the same connections with the new branch as they had with the old one, and similarly the new branch has the same connection with its new trunk as it had with its previous trunk.

Picking corresponds to simply taking a branch from a given individual and copying it in the genome of another individual, without removing any material. This is our only addition to Sims' genetic operators, but it seems to have an impact on the continuous appearance of complex features.

Mutation occurs by modifying each parameter of the genome with a certain probability P_{mut} (in this paper, $P_{mut} = 0.04$). The mutation operator proceeds in a sequential manner. First, with probability P_{mut}, a random limb may be deleted from the genome (with the restriction that no creature may have less than two limbs). Then a new randomly generated limb (with randomly generated neural information) may be created. Then, a recursive cycle may be unrolled (if there is a recursion in the genome). Then each slot in the neural array may be "flipped" (i.e. empty slots are filled with a new neuron with randomly assigned connections, existing neurons are deleted). Each interneuron may be turned into a sensor, and vice versa (proprioceptors and actuators are fixed for all creatures). Each sensor neuron can change types (i.e. from sensing the box to sensing the opponent and vice versa). Then, the threshold value of each existing neuron may be modified through gaussian perturbation (standard deviation 0.4) within the $[-1,1]$ torus. The output function may be changed. Each connection of every existing neuron may be "flipped", i.e. created (and randomly assigned) if it is unassigned, or deleted otherwise. The weight of each existing connection may then be modified through gaussian perturbation (standard deviation 0.4) within the $[-1,1]$ torus. The source of each connection (i.e. the neuron from which it originates) may be randomly reassigned. Finally, with probability P_{mut}, morphological information for each node is mutated. Morphological mutation performs one randomly selected operation out of seven possibilities: reassigning a given limb to a different "ancestor" limb (which amounts to moving a whole branch along the organism), randomly assigning a new length, width or height to the limb, modifying either of its orientation angles (possi-

ble orientations are discrete multiples of $\pi/4$; mutation occurs by choosing a new value within the range $[-\pi/2, \pi/2]$ around the current value), switching the orientation of its joint (horizontal or vertical), and flipping its "reflection" flag. Again, each of these modifications is applied with probability P_{mut} for each parameter.

Experiments and Results

The task being considered is the "box-grabbing" contest described in (Sims, 1994a). Two creatures compete to gain control of a cubic box. At the beginning of each contest, the box is placed at the centre of the environment. Both competing creatures are placed on opposite sides of the box, at a certain distance from it. As in Sims, creatures are pushed behind a diagonal plane slanted by 45 degrees so they cannot gain an undue advantage by their height. Both creatures are then left to act for a given period of time. At the end of the evaluation period, the score for each creature is determined in the following way: if one creature is at distance $d1$ from the box (as defined by the distance between the centre of the creatures' closest limb to the box, and the centre of the box) and the other at distance $d2$, then the former creature's score is $d1 - d2$, while the latter's is $d2 - d1$. Creatures have four kinds of sensors, measuring the x and y distances of either the opponent or the box, within the frame of reference of the limb in which the sensor exists.

The evolutionary algorithm that we use is a "3-strikes-out" algorithm, a simple steady-state genetic algorithm for coevolution of our devising. The simplest description of this algorithm, in a one-population, symmetric competition context, is as follows: run competitions between randomly chosen individuals, keep track of each individual's victories and defeats, and replace any individual that has accumulated 3 defeats. Our preferred reproduction strategy, which we use in the present experiments, is simply to replace the removed individual with either a heavily mutated copy of itself, or a heavily mutated copy of the individual by which it has just been defeated, or some form of recombination between both.

When the problem involves two genetically separate populations (say, A and B), 'victories' and 'defeats' must occur between fellow members of the same population (even though simulated competitions still occur between members of different populations). Victory or defeat is determined by comparing the results of two individuals against one single opponent of the other population. This involves two simulated competitions; however, the result of one of these competitions is used in the next round (when two individuals of the other population are compared), so each new comparison requires the simulation of just one new competition.

Our experiments involved two populations of 100 individuals each. Interactions between two individuals were simulated over 15000 timesteps, each timestep corresponding to 0.01 second of simulated time. Some examples can be seen in Fig. 2. Others are available at the URL mentioned in the introduction of this paper.

From visual inspection it is clear that our system managed to come up with diverse solutions to the problem. Locomotive behaviours emerged rather easily. A locomotive behaviour requires some sort of cyclic, oscillating movement, which in turn implies some coordination between proprioceptors and actuators (Miconi and Channon, 2005). Additionally, sensorimotor coordination was observed, for example the creature in the third picture of Fig. 2 can adjust its trajectory depending on the position of the box.

Comparing these results to those obtained by Sims is not easy for several reasons. First, Sims apparently used far fewer interactions (100 generations with two populations of 100 individuals, which is much less than the $\tilde{1}$00K evaluations used in the present experiment). Furthermore Sims used complex neurons which automatically provided behaviours such as oscillations 'for free'. Restrictions imposed by Sims on the morphology of creatures (maximum numbers of genes, of blocks, etc.) are unknown.

The diversity of morphological plans is reflected in the neural controllers. Figure 3 describes the functional subnetworks of two creatures. The network on the left corresponds to one appendage of a crawling creature (the other appendage, being a symmetric replication, contains a similar network). It is easy to notice the feedback loop between the proprioceptors and actuators of limbs 1 and 2, leading to an oscillating movement of the corresponding joint which propels the creature forward. Other, small-weight connections originating from sensors lead to a folding of the appendage in the direction of the box.

The network on the right, however, is more complicated. It describes the functional subnetwork of the "encircling" creature shown in Fig. 2. The basic core of the neural controller is easily isolated: the "feedback cascade" in limbs 2, 3 and 4 provides the coordinated sinusoidal movement which propels the creature forward (limbs 2, 3 and 4 are a recursive replication sequence which corresponds to the segmented motile appendage of the creature). The sensor in limb 1 influences the shape of the creature in order to change the direction of motion depending on the relative position of the box.

However the behaviour of the creature is also dependent on all other connections and neurons, even though their exact function is difficult to isolate individually. Collectively, they ensure that the creature assumes the right shape and posture (and maintains them) to follow the desired trajectory. This occurs through coordination of activities which are tuned to the morphology of the creature (mass and orientation of limbs, etc) in order to skew the trajectory correctly. Experimental lesions in this network reduce the efficiency of the creature in various ways, which range from a slowdown to a loss of trajectory. Several other connections were present in the network (including between the neurons depicted) but were found to have no impact on the behaviour

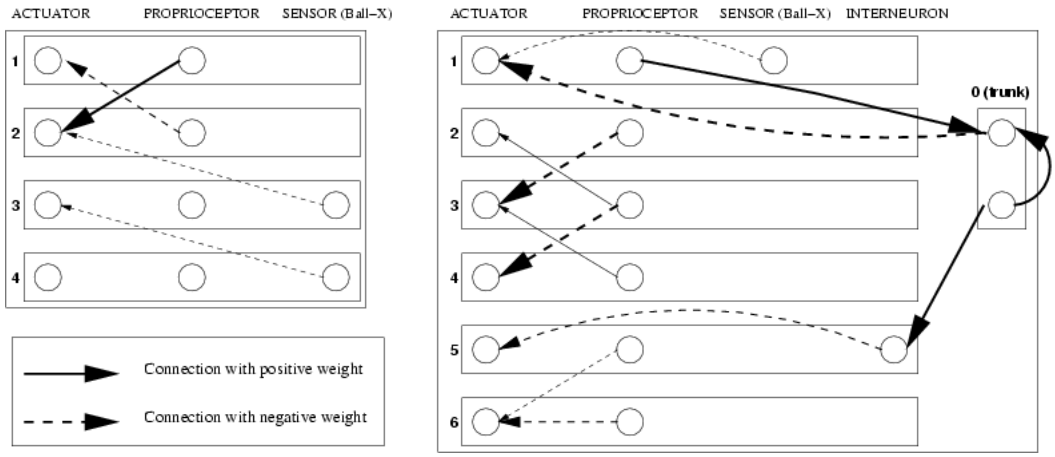

Figure 3: Functional subnetworks extracted from the neural networks of two creatures. The network on the left is taken from one of the accordion-shaped appendages of the creature seen in the third picture of Fig. 2. The network on the right is taken from the "encircling" creature seen in the fourth picture of Fig. 2. Thickness of lines is proportional to connection weight. See text for details.

of the creature. All in all, it appears that an intricate collection of characters (neurons, connections, shapes of limbs) with no obvious individual effect, collectively provide a useful and reliable function - a good example of evolutionary "bricolage" (tinkering).

Acknowledgements

This research is now being funded by the School of Computer Science at the University of Birmingham, and was previously funded by the Intelligent Systems Group in the Department of Electronic and Computer Engineering at the University of Portsmouth, under the direction of Dr David Brown.

References

Bongard, J. C. and Pfeifer, R. (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In (Spector et al., 2001), pages 829–836.

Hornby, G. S. and Pollack, J. B. (2001). Body-brain coevolution using L-systems as a generative encoding. In (Spector et al., 2001), pages 868–875.

Komosinski, M. (2000). The world of framsticks: simulation, evolution, interaction. In *Procs of 2nd International Conference on Virtual Worlds (VW2000), Paris*, pages 214–224. Springer-Verlag (LNAI 1834).

Mesot, B. (2004). Self-organisation of locomotion in modular robots: A case study. Master's thesis, EPFL, Lausanne.

Miconi, T. and Channon, A. (2005). A virtual creatures model for studies in artificial evolution. In *IEEE Congress on Evolutionary Computation (CEC 2005)*.

Ray, T. S. (2000). Aesthetically evolved virtual pets. In Maley, C. C. and Boudreau, E., editors, *Artificial Life 7 Workshop Procs*, pages 158–161.

Sims, K. (1994a). Evolving 3d morphology and behavior by competition. In Brooks, R. and Maes, P., editors, *Procs 4th Intl Works on Synthesis and Simulation of Living Systems (ALIFE IV)*, pages 28–39. MIT Press.

Sims, K. (1994b). Evolving virtual creatures. In *SIGGRAPH 94*, pages 15–22. ACM Press.

Spector, L., Goodman, E. D., Wu, A., and Langdon, W. B., editors (2001). *Procs GECCO 2001*. Morgan Kaufmann.

Taylor, T. and Massey, C. (2001). Recent developments in the evolution of morphologies and controllers for physically simulated creatures. *Artificial Life*, 7(1):77–87.