Evolution and Development of a Central Pattern Generator for the Swimming of a Lamprey

Auke Jan Ijspeert*
Department of
Artificial Intelligence
University of Edinburgh
5 Forrest Hill
Edinburgh EH1 2QL, U.K.
aukei@dai.ed.ac.uk

Jérôme Kodjabachian
AnimatLab
OASIS-LIP6
Université Pierre et
Marie Curie
4 place Jussieu
F-75005 Paris, France
Jerome.Kodjabachian@
poleia.lip6.fr

Abstract This article describes the design of neural control architectures for locomotion using an evolutionary approach. Inspired by the central pattern generators found in animals, we develop neural controllers that can produce the patterns of oscillations necessary for the swimming of a simulated lamprey.

This work is inspired by Ekeberg's neuronal and mechanical model of a lamprey [11] and follows experiments in which swimming controllers were evolved using a simple encoding scheme [25, 26]. Here, controllers are developed using an evolutionary algorithm based on the SGOCE encoding [31, 32] in which a genetic programming approach is used to evolve developmental programs that encode the growing of a dynamical neural network. The developmental programs determine how neurons located on a two-dimensional substrate produce new cells through cellular division and how they form efferent or afferent interconnections. Swimming controllers are generated when the growing networks eventually create connections to the muscles located on both sides of the rectangular substrate. These muscles are part of a two-dimensional mechanical simulation of the body of the lamprey in interaction with water.

The motivation of this article is to develop a method for the design of control mechanisms for animal-like locomotion. Such a locomotion is characterized by a large number of actuators, a rhythmic activity, and the fact that efficient motion is only obtained when the actuators are well coordinated. The task of the control mechanism is therefore to transform commands concerning the speed and direction of motion into the signals sent to the multiple actuators. We define a fitness function, based on several simulations of the controller with different commands settings, that rewards the capacity of modulating the speed and the direction of swimming in response to simple, varying input signals. Central pattern generators are thus evolved capable of producing the relatively complex patterns of oscillations necessary for swimming. The best solutions generate traveling waves of neural activity, and propagate, similarly to the swimming of a real lamprey, undulations of the body from head to tail propelling the lamprey forward through water. By simply varying the amplitude of two input signals, the speed and the direction of swimming can be modulated.

Keywords
neural control, genetic programming, developmental encoding, SGOCE, simulation, central pattern generator, swimming, lamprey

* Current address: Brain Simulation, & Computational Learning and Motor Control Labs, Department of Computer Science, Hedco Neurosciences Bldg., University of Southern California, Los Angeles, CA 90089. E-mail: ijspeert@rana.usc.edu.

1 Introduction

This article presents an experiment in which dynamical neural networks are developed for controlling the locomotion of a simulated fish. The experiment takes inspiration from biology at three levels: the type of locomotion, the type of control mechanisms, and the type of design method. Its aim is to develop a method for the automatic generation of control mechanisms for animal-like locomotion. It can also be considered as an experiment in computational neuroethology [1, 7], in the sense that we study how a network of neurons can generate a (simple) behavior, locomotion, in a (simple) environment, water.

Most animal locomotion is fascinating by its agility and energy efficiency. Using muscles as basic actuators, animals can fly, run, swim, and so forth with great dexterity. This allows them to move easily in environments in which man-made robots have significant difficulties. As wheeled robots, for instance, are strongly limited in the kind of environments in which they can move, some engineers have turned to biological systems for inspiration and developed robots that move using more animal-like types of gaits [4, 8]. The means of locomotion are then more complex than powered wheels and involve a greater number of actuators generally used in a rhythmic way. Biological types of locomotion require therefore complex control mechanisms, as motion is obtained only when the different actuators are well coordinated.

The control mechanisms in animals are provided by networks of neurons. For many animals, both vertebrate and invertebrate, the control of locomotion is distributed and the patterns of oscillations are produced by circuitries called *central pattern generators* (CPGs) [10, 17, 19]. These circuits can produce the different patterns of oscillations necessary for locomotion without oscillating input from the brain or from sensory feedback. In the cat, for instance, the walking gait can be induced by sending a simple signal to the brain stem. Different gaits can then be obtained by increasing the amplitude of the signal, with the gait passing from walking to trotting to running [18]. A CPG therefore provides the template of neural activity necessary for locomotion that can be modulated, when necessary, by higher control and by sensory feedback depending on the external conditions.

The bodies and nervous systems of animals are the result of two adaptation phenomena: natural evolution and development. In this work, we will design neural controllers for locomotion using a method that takes inspiration from these phenomena in the form of two abstract ideas: (a) an evolutionary algorithm that creates potential solutions by breeding and mutating other solutions and that selects the best solutions according to a fitness criterion, and (b) a representation that encodes solutions through a developmental program containing the rules of how a solution is grown. Evolutionary algorithms are now commonly used for designing neural networks [37, 48], and several developmental encoding schemes have recently been designed (see [30] for a review).

We study in particular the swimming of the lamprey. A lamprey is a fish that propels itself in water by undulation of its body, without the use of fins. The undulation is a traveling wave propagating from head to tail, with a constant wavelength along the spinal cord. The neural circuitry for locomotion of the lamprey has been studied extensively by neurobiologists (see [20] for a review), who developed models of the CPG whose capacity to reproduce most of physiological observations has been demonstrated through simulation [13, 45]. In previous work [25, 26], we showed that a genetic algorithm with a simple direct encoding scheme and a staged-evolution approach could be used to design a connectionist model of the CPG of the lamprey with several similarities with the biological circuitry. That work was inspired by Ekeberg's

connectionist model of the biological CPG and his mechanical simulation of the lamprey [11] and investigated the space of possible neural configurations for the control of anguiliform locomotion. We showed, in particular, that a variety of architectures could produce patterns of oscillations for anguiliform swimming, several of them showing a better performance than the biological configuration (in terms of ranges of frequencies, of oscillation, and of speeds of swimming that they can produce in the simulation). We also demonstrated that the genetic algorithm could be useful in neurobiological modeling as a tool for instantiating synaptic weights of a circuitry with known connectivity. Ekeberg's handcrafted model could thus be improved to better fit physiological observations.

The work presented here is less close to neurobiology and concentrates on developing a method for the design of control mechanisms for animal-like locomotion. The motivation is to generate automatically a neural controller given a high-level characterization of the desired locomotion gait in terms of observable characteristics such as the speed of locomotion or the capacity to change direction. Compared to [25, 26], the new aspects are that (a) simpler dynamical neuron models are used, (b) controllers are evolved in a single stage with a fitness function that is only based on observable behaviors of the mechanical simulation without taking into consideration the neural activity, and (c) controllers are evolved using a genetic programming approach with a developmental encoding, SGOCE (simple geometry oriented cellular encoding) [31, 32]. With SGOCE, developmental programs are evolved that determine how neurons located on a two-dimensional substrate produce new cells through cellular division and how they form efferent or afferent interconnections. That method has been successfully used to develop neural controllers for the walking of a virtual six-legged insect and for behaviors, such as gradient following and obstacle avoidance [31, 32]. The task here is to generate neural controllers for swimming given a fixed body structure of the lamprey and given a high-level characterization of the desired behavior defined by the fitness function. Through two input signals applied to the network we want to be able to initiate swimming and to modulate the speed and the direction of motion by simply varying the amplitude of these signals. The evolved controllers could therefore easily be used by higher control centers for sensorimotor coordination experiments of a lamprey-animat.

Swimming controllers have been evolved by other researchers. In [42, 43], for instance, the parameters of algorithmic controllers are evolved for the swimming of simulated swimming creatures. Another example is Sims' evolution of swimming controllers composed of "neural nodes" that represent simple mathematical functions such as sum, product, sine, cosine, and so on. [38]. In [40, 41], simulated annealing is used to develop algorithmic swimming controllers that minimize energy consumption as well as the final distance from a target location, for instance. Finally, algorithmic controllers for anguiliform swimming were also developed using self-organizing rules (adaptive ring rules) [34]. The research presented here differs from these works in that we develop controllers implemented as dynamical neural networks. There are several motivations behind this choice: First, developing potential neural controllers for swimming may give some insights into the functioning of corresponding biological control systems; second, neural networks present interesting properties such as rich dynamics, distributed control, and robustness against noise; and, finally, parameters of neural networks such as synaptic weights and time constants are ideal low-level primitives on which to apply artificial evolution. Similar considerations have led numerous researchers to combine neural networks and evolutionary algorithms for the control of animal-like locomotion, in particular legged locomotion [3, 9, 22, 31, 35]. Except for a few exceptions [32], these works, however, only considered the problem of pattern generation, without considering how to control these patterns for modulating the locomotion, namely the speed

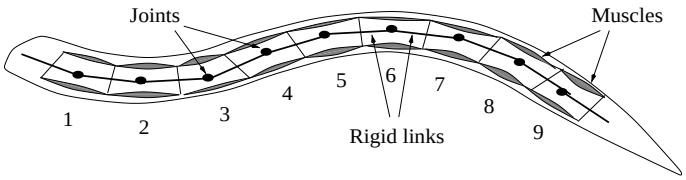


Figure 1. Schema of the mechanical simulation, as developed in [11]. The body is composed of 10 rigid links connected through nine one-degree-of-freedom joints. The torque of each joint is determined by the contraction of two muscles fixed in parallel.

and the direction of movement. As mentioned earlier, this last point is one of the main motivations of this article.

In the next sections we describe the neuronal and mechanical model of the lamprey (Section 2), the developmental encoding scheme (Section 3), the evolutionary algorithm (Section 4), the fitness function (Section 5) and the results of 10 evolutions (Section 6). The method and the results are further discussed in Section 7.

2 Neuronal and Mechanical Model of the Lamprey

The CPG of the lamprey has been modeled at several levels of abstraction. Models based on biophysical simulations of neurons have proved that the current knowledge of the CPG circuitry and the type of interneurons involved is sufficient to produce simulations that agree with the physiological observations [13, 45]. At a more abstract level, connectionist models have been made that demonstrate that the connectivity in itself can produce most of the observed patterns of oscillations without the need for complicated neuronal mechanisms [5, 11, 46]. Finally, at the most abstract level, the CPG has been modeled as a chain of abstract oscillators for the study of interoscillator coupling [33, 47].

In this work, we develop connectionist models, that is, models composed of neurons of intermediate complexity between abstract binary neurons used traditionally in artificial neural networks and detailed compartmental models used in computational neuroscience. Instead of simulating each activity spike of a real neuron, a neuron unit is modeled as a *leaky-integrator* that computes the average firing frequency [23]. According to this model, the mean membrane potential m_i of a neuron N_i is governed by the equation:

$$\tau_i \cdot dm_i/dt = -m_i + \sum w_{i,j} x_j$$

where $x_j = (1 + e^{(m_j + b_j)})^{-1}$ represents the neuron's short-term average firing frequency, b_j is the neuron's bias, τ_i is a time constant associated with the passive properties of the neuron's membrane, and $w_{i,j}$ is the synaptic weight of a connection from neuron N_j to neuron N_i. Each neuron exhibits an internal dynamics and even small networks of these neurons have proven able to produce rich dynamics [2].

The network of neurons determines the muscular activity of a simple mechanical simulation of the body of the lamprey in interaction with water (Figure 1). The two-dimensional simulation used in this work is based on that developed by Ekeberg [11]. The body of the lamprey is simulated as 10 rigid links connected by one-degree-of-freedom joints. Two muscles are connected in parallel to each joint and are modeled as a combination of springs and dampers. Muscles can be contracted when they receive an input; their spring constant is then increased proportionally to the input, thus reducing the resting length. The motion of the body is calculated from the accelerations of the

Table 1. Set of instructions used in the developmental programs.

Instruction	Definition
DIVIDE α r	create a new cell
GROW α r w	create a connection to another cell
DRAW α r w	create a connection from another cell
SETBIAS b	modify the bias parameter
SETTAU τ	modify the time constant parameter
SETINPUTW w	modify the input weight
DIE	trigger cellular death

links. These accelerations are due to three forces: the torque due to the muscles, the inner forces linked with the mechanical constraints keeping the links together, and the forces due to the water (for more details, see [11, 24]).

3 Developmental Encoding Scheme

Swimming controllers are evolved using an encoding scheme that relates the animat's genotype—a developmental program—and phenotype—a dynamical neural network; and an evolutionary algorithm that breeds and selects the developmental programs within a space defined by some syntactic constraints.

We use SGOCE [31, 32] for encoding the controllers. This encoding scheme is a geometry-oriented variation of Gruau's cellular encoding [22]. In the scheme, developmental programs are evolved that determine how neurons divide and become connected to each other on a two-dimensional metric substrate. The substrate contains precursor cells that will develop into sets of neurons, and nodes that correspond to inputs (the control signals) and outputs (the signals sent to the muscles).

Each cell within the control architecture is assumed to hold a copy of the developmental program. The developmental program has a treelike structure like those of genetic programming. The nodes in the tree belong to a small set of developmental instructions (see Table 1) or are structural instructions added to help define the grammar (see below). A local coordinate frame is attached to each precursor cell (Figure 2). A cell division instruction (DIVIDE) makes it possible for a given cell to generate a copy of itself. A direction parameter (α) and a distance parameter (r) associated with that instruction specify the position of the daughter cell to be created in the coordinates of the local frame attached to the mother cell. Then, the local frame associated to the daughter cell is centered on that cell's position and is oriented as the mother cell's frame. Each time a cell divides, its daughter cell receives a copy of its afferent and efferent connections. Two instructions (GROW and DRAW) respectively create one new efferent and one new afferent connection. The cell or the input-output node to be connected to the current one is the closest to a target position that is specified by the instruction parameters, provided that the target position lies on the substrate. No connection is created if the target is outside the substrate's limits. The synaptic weight of the connection is given by the parameter w. Two additional instructions (SETTAU and SETBIAS) specify the values of a neuron's time constant τ and bias b. Finally the instruction DIE causes a cell to die. In addition to the connections created by the developmental program, we cause each created neuron to draw automatically an afferent connection from the input node of the side of the precursor cell it stems from. The synaptic weight of that connection can be modified by the SETINPUTW instructions. An example of how a developmental program is decoded into the corresponding set of neurons is given in Figure 2.

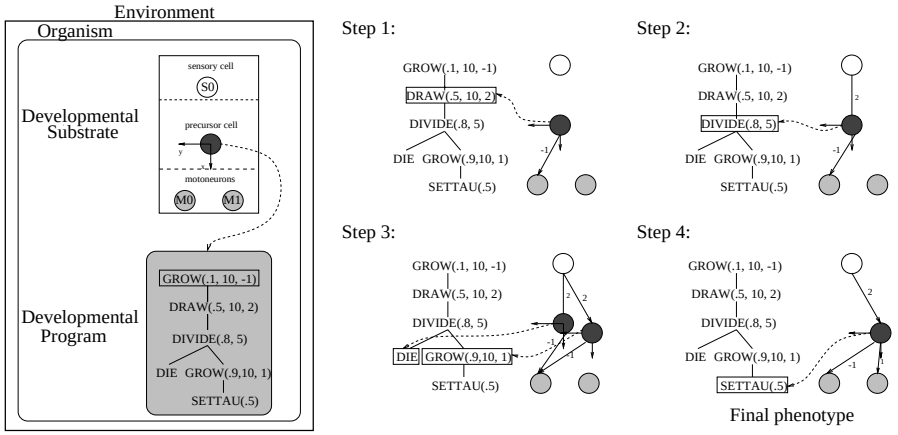


Figure 2. Example of the decoding of a developmental program into a set of neurons. First, a connection is grown by the precursor cell to an output cell (Step 1). Second, another connection is drawn by this precursor cell from the input cell (Step 2). In both cases, the cell with which the connection is made is the closest one to a target point whose polar coordinates are given in the local frame of the cell by the instruction's parameters. At the next developmental step (Step 3), the precursor cell creates a daughter cell that reads the right subnode of the DIVIDE instruction while the mother cell reads the left subnode and is killed, because the corresponding instruction is a DIE instruction. Note that the DIVIDE instruction has duplicated the connections of the initial cell. Finally, the daughter cell grows a new connection to another output cell (Step 4) and modifies the value of its time constant parameter. After the development, this cell is considered as a neuron with its specific connections and parameters.

We use a substrate that corresponds to the body of the lamprey (Figure 3). Nine output nodes are fixed on each side of the body. These nodes are connected to the muscles such that the signals sent to them determine the contraction of the muscles. We also fix the position of 18 precursor cells, located symmetrically on both sides of the body. The substrate can therefore be seen as made of nine segments, with two output nodes and two precursor cells each, corresponding to the nine joints of the mechanical simulation. Each of the 18 precursor cells holds a copy of the same developmental program. However the local frame attached to each cell need not have the same orientation, and in particular we make the precursor cells on each side (left and right) of the body have opposite orientations, which forces the developed controllers to be symmetrical. The two input nodes are also included in the substrate, but as each neuron automatically has an afferent connection from them, their exact position is not

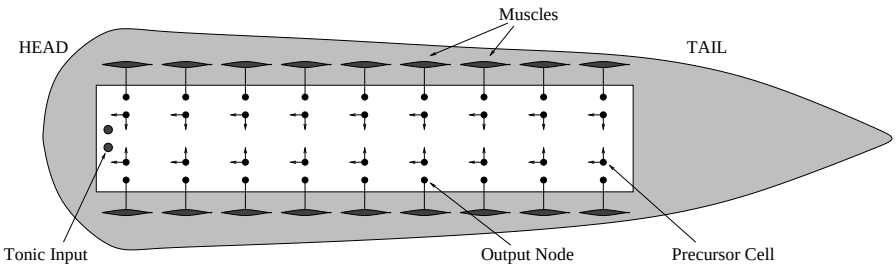


Figure 3. Substrate of the developmental process. The rectangular substrate extends over the length of the mechanical body. It contains 18 output nodes located symmetrically on both sides of the substrate that determine the contraction state of the muscles (Figure 1). It contains also two input nodes that provide the left and right tonic input of the CPG. Eighteen precursor cells are spread over the substrate. Left and right precursor cells have local frames that are oriented with a left-right symmetry. As all precursor cells develop using the same developmental code, this means that a left-right symmetry of the developed network is automatically created.

Table 2. Grammar used to restrict the trees that can be evolved. The production rules organize the development into several stages that schedule the development and restrict the maximum number of neurons.

Terminal symbols
DIVIDE, GROW, DRAW, SETBIAS, SETTAU, SETINPUTW, DIE, NOLINK, DEFBIAS, DEFTAU, DEFINPUTW, SIMULT4
Variables
Start1, Level2, Neuron, Bias, Tau, InputW, Connex, Link
Production rules
Start1 $\longrightarrow$ DIVIDE(Level2, Level2)
Level2 $\longrightarrow$ DIVIDE(Neuron, Neuron)
Neuron $\longrightarrow$ SIMULT4(Bias, Tau, InputW, Connex) | DIE
Bias $\longrightarrow$ SETBIAS | DEFBIAS
Tau $\longrightarrow$ SETTAU | DEFTAU
InputW $\longrightarrow$ SETINPUTW | DEFINPUTW
Connex $\longrightarrow$ SIMULT4(Link, Link, Link, Link)
Link $\longrightarrow$ GROW | DRAW | NOLINK
Starting symbol
Start1

important. Note that connections created by the GROW and DRAW instructions are not limited to neurons issued from the same precursor cell but these instructions can potentially connect any cell on the substrate within a fixed range that corresponds to the maximal value of the parameter r of the GROW and DRAW instructions. This means that networks can in principle be evolved with paths of connections between all neurons of the substrate. In the present article, the range of connection is limited such that only connections between neurons of neighboring segments can be developed.

To reduce the size of the genetic search space and the complexity of the generated networks, a context-free tree grammar is used to force each evolvable program to have the structure of a well-formed tree (Table 2). The set of terminal symbols consists of the developmental instructions listed in Table 1 and of additional *structural instructions* that have no side effect on the developmental substrate. The grammar of Table 2 permits the scheduling of the development to be organized into several marked stages. First, cell divisions will occur and the connections from the command nodes to the precursor cells will be copied. Second, the different cells created will either die or modify their time constant, bias, and input weight parameters. Finally, each surviving cell will create connections with its surrounding cells. As no more than two successive cell divisions can occur, this grammar limits the total number of neurons in a controller to 72 neurons.

4 Evolutionary Algorithm

We used the SGOCE encoding with a steady-state genetic algorithm. To slow down convergence and to favor the apparition of ecological niches, the algorithm involves a population of N randomly generated, well-formed developmental programs distributed over a circle. Its functioning is sketched in Figure 4. The same algorithm was used for the evolution of legged-locomotion controllers [31, 32].

The following procedure is repeated until a given number of individuals have been generated and tested:

1. A position P is chosen on the circle.

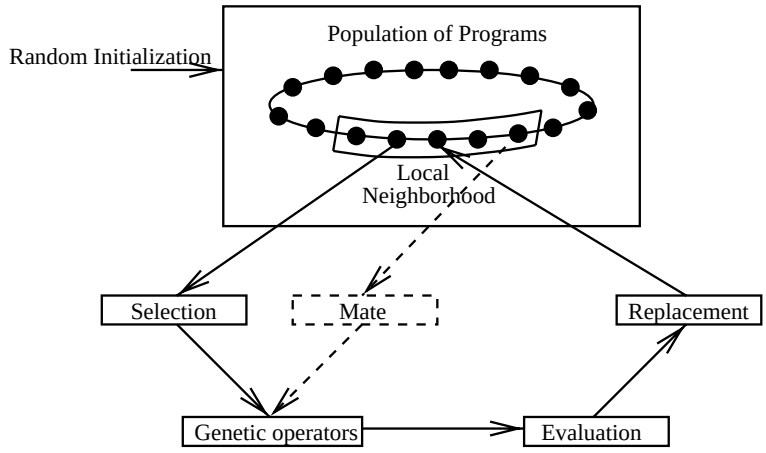


Figure 4. The evolutionary algorithm.

2. A two-tournament selection scheme is applied, in which the best of two programs randomly selected from the neighborhood of P is kept (more details can be found in [27]).
3. The selected program is allowed to reproduce and three genetic operators possibly modify it. The first operator, the recombination operator, is applied with probability p_c. It exchanges two *compatible* subtrees, that is, subtrees that can be derived from the same grammatical rule, between the program to be modified and another program selected from the neighborhood of P. Two types of mutation are used. The first mutation operator is applied with probability p_m. It changes a randomly selected subtree into another compatible, randomly generated one. The second mutation operator is applied with probability 1. It modifies the values of a random number of parameters, implementing a *constant perturbation strategy* [39]. The number of parameters to be modified is drawn from a binomial distribution $B(n, p)$.
4. The fitness of the new program is assessed by collecting statistics while the swimming of the lamprey mechanical model, controlled by the corresponding artificial neural network, is simulated over a given period of time (see below).
5. A two-tournament anti-selection scheme, in which the worse of two randomly chosen programs is selected, is used to decide which individual (in the neighborhood of P) will be replaced by the modified program.

In all the experiments reported here $p_c = 0.6$, $p_m = 0.2$, $n = 6$, and $p = 0.5$.

5 Fitness Function

A developmental program is given a fitness value that depends on the capacity of its corresponding network to control swimming efficiently. Our aim is to develop controllers that can produce patterns of oscillations necessary for swimming when receiving tonic (i.e., nonoscillating) input, and that can modulate the speed and the direction of swimming when the amplitude of the left and right control signals are varied.[1]

1 For reasons of simplicity, a measure of muscle effort is not taken into account in this fitness function. In future work, it would be interesting not only to optimize the controllability of the swimming gait, but also to minimize its energy consumption.

Table 3. Variables and boundaries for the fitness function. A contortion is measured as the passage of the center of the body from one side to the other of the line between head and tail. The speed range is measured relative to the maximum speed.

Function	Variable	[*bad*,*good*] Boundaries
fit_contortion	Number of contortions	[0,10] contortions
fit_max_speed	Maximum speed	[0.0,1.0] m/s
fit_speed_range	Relative speed range	[0.0,1.0]
fit_turning	Deviation angle	$[0.0,0.75\pi]$

We therefore define a fitness function, based on observable behaviors in the mechanical simulation, rewarding controllers that

1. produce contortions, where a contortion is defined as the passage of the center of the body from one side to the other of the line between head and tail
2. reach high swimming speeds
3. can modulate the speed of swimming when the tonic input is varied, with the speed increasing monotonically with the level of input
4. can change the direction of swimming when an asymmetrical tonic input (between the two sides of the body) is applied

The evaluation of a developmental program consists of a series of simulations of the corresponding developed controller with different commands settings. Fixed-time simulations (6,000 ms) are carried out with different levels of tonic input to determine the range of speeds the controller can produce. Asymmetrical initial states for the neurons are created by applying an asymmetrical input during the first 50 ms of the simulation. Starting from a fixed level of input (1.0), the input is varied by increasing and decreasing steps of 0.1 and the range of speed in which the speed increases monotonically with the tonic input is measured.[2] If the range of speeds includes a chosen[3] speed (0.15 m/s), a simulation is performed (at the tonic level corresponding to that speed) with a short period of asymmetric tonic input to evaluate the capacity of the controller to induce turning.[4] Turning is evaluated by measuring the deviation angle between the directions of swimming before and after the asymmetric input.

The mathematical definition of the fitness function is the following:

$$\text{fitness} = \text{fit\_contortion} \cdot \text{fit\_max\_speed} \cdot \text{fit\_speed\_range} \cdot \text{fit\_turning} \in [(0.05)^4, 1.0]$$

where *fit_contortion, fit_max_speed, fit_speed_range*, and *fit_turning* are functions that are limited between 0.05 and 1.0 and that vary linearly between these values when their corresponding variables vary between two boundaries, a *bad* and a *good* boundary. The variables for each function and their corresponding boundaries are given in Table 3. If the speed range does not include the chosen speed of 0.15 m/s, the turning ability is not measured and *fit_turning* = 0.05. The fact that the fitness function is a product rather than a sum ensures that controllers that perform equally in all four aspects will be more favored compared to controllers performing well in some aspects but not in others.

2 The speed of swimming is measured as the average speed during the second half of the simulation.

3 This value corresponds to approximately half the maximum speed reached by the best solutions.

4 The sequence of control signals for the turning evaluation is symmetric left and right signals for 3,000 ms, asymmetric (±10% between left and right) for 1,000 ms, and symmetric for the last 2,000 ms.

Table 4. Results of the evolutions of central pattern generators. See text for details.

Run	Fitness	Speed [m/s]	Frequency [Hz]	Relative Lag [%]
1	0.00	[0.00, 0.08]	–	–
2	0.00	[0.00, 0.11]	–	–
3	0.01	[0.00, 0.11]	–	–
4	0.14	[0.00, 0.15]	–	–
5	0.24	[0.00, 0.16]	[1.16, 1.60]	–
6	0.27	[0.06, 0.27]	[0.46, 2.50]	–
7	0.01	[0.00, 0.12]	[0.69, 1.75]	[0.0, 0.0]
8	0.05	[0.02, 0.16]	[0.94, 0.95]	[0.0, 8.2]
9	0.24	[0.00, 0.27]	[0.57, 2.31]	[1.5, 11.2]
10	0.49	[0.09, 0.47]	[1.24, 5.38]	[16.0, 20.7]

6 Results

6.1 Evolutions

We carried out 10 evolutions with populations of 50 controllers. Each run started with a different random population. A run was allowed to evolve for 100 generations (5,000 tournaments), taking approximately 450 CPU hours on an Ultra 1 Model 140s SUN station. Table 4 gives the performances of the best evolved controllers of each run (controllers 1 to 10 respectively). The evolved controllers can be classified into three categories: those that do not produce oscillations (controllers 1, 2, 3, and 4), those that produce chaotic behavior (controllers 5 and 6) and those that produce stable oscillations (controllers 7, 8, 9, and 10). Controllers 1, 2, 3, and 4 do not produce oscillations. For these controllers, the given speeds indicate the range of speeds that can be reached through the initial thrust after 6,000 ms. Controllers 5 and 6 produce unstable oscillations, that is, oscillations without a constant period. The given frequencies correspond therefore to the average (minimal and maximal) frequencies and the given speeds correspond to minimal and maximal speeds, although the speed does not depend monotonically on the excitation. Also, because the shape of the signals is too variable, the relative lag is not given. The controllers 7, 8, 9, and 10 produce stable oscillations. For these controllers, the speed, frequency, and lag values correspond to the minimal and maximal values obtained within the range of excitations for which the speed depends monotonically on the excitation. Figure 5 shows the swimming behavior they induce. We describe next the main characteristics of the evolved controllers; for a more detailed analysis see [27].

6.2 Nonoscillating Networks

Four evolutions converged to nonoscillating solutions (controllers 1,2,3, and 4). These controllers do not have the circuitry to sustain oscillations but still manage to propel the body forward by creating a few contortions that give the body an initial thrust. These contortions are due to the delayed transitions of the neurons from their initial state to their equilibrium state. The delays in the transitions lead to delays in the contraction of the muscles, which happen to propel the body forward. Once the equilibrium states are reached, the contortions of the body cease and the lamprey eventually comes to a standstill. As the duration of the simulations was limited, these controllers were rewarded for their remaining speed at the end of the simulation.

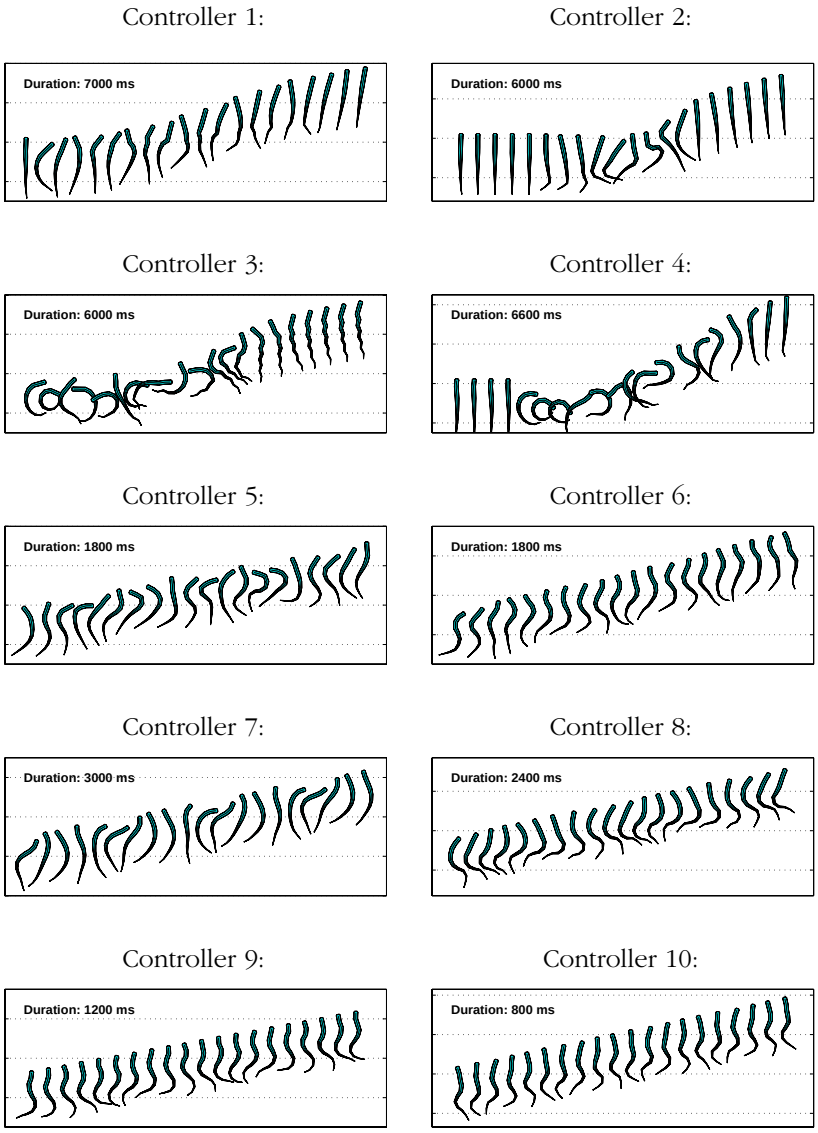


Figure 5. Swimming produced by controllers 1 to 10. The horizontal lines are separated by 200 mm.

6.3 Chaotic Networks

Two evolutions converged to solutions producing chaotic oscillations with variable cycle duration and variable signal shapes (controllers 5 and 6). As the phase lag between segments is on average positive (i.e., the neural activity travels from head to tail), these solutions still manage to propel the lamprey forward but with a speed and a direction of swimming that are not constant. Because of these variations of speed, these solutions do not present a monotonic relation between the tonic excitation and the speed. However they received a high fitness value because the speeds measured in the evaluation function happened to increase monotonically for the excitation steps tested. The nonmonotonicity is revealed when the control of speed is analyzed with smaller excitation steps than those used for the evaluation. Note that it is not surprising

that chaotic oscillators have been evolved, as systems made of as few as three interconnected neurons can already present chaotic dynamics with this mathematical model of a neuron [2].

6.4 Oscillating Networks

Four evolutions converged to solutions producing permanent and stable oscillations, with all segments being phase locked and oscillating at the same frequency (controllers 7, 8, 9, and 10). The phase relation between segments varies, however, from one controller to the other, as do their configurations, which vary in the number of oscillators for producing the oscillatory activity. By oscillator, we mean the circuits that could potentially oscillate by themselves when isolated from the rest of the network (provided that the simulated neurons receive adequate tonic input). It can be shown that the smallest oscillator with this model of neurons is composed of two neurons with self connections [2].

Controller 7 has the peculiarity of being made of two separate chains of oscillators on both sides of the body that are connected through excitatory contralateral connections (data not shown, see [27]). Neurons on the same side oscillate in synchrony, and, because neurons on contralateral sides belong to different oscillators, the contralateral phase relation depends on the initial conditions. With the asymmetrical initial conditions of the evaluation function, the body produces a "C" bending, but with a contralateral phase relation that is not perfectly out of phase. Because of the different inertial forces between head and tail segments, this results in the body making an asymmetrical traveling undulation with a large wavelength, which slowly propels the body forward (see Figure 5). If the simulation is started with nearly symmetrical initial conditions, left and right muscles contract in phase and the lamprey does not progress.

Controller 8 is composed of one oscillator per segment interconnected over the spinal cord (data not shown, see [27]). Each oscillator is distributed over both sides through inhibitory contralateral connections, and the left and right neurons of each segment are therefore perfectly out of phase. The phase relation between segments is slightly negative between segments closest to the head and gradually increases to become positive toward the tail. This means that the muscle activity travels forward to the head and backward to the tail from a segment in the middle of the trunk. As the negative phase difference between the first segments is small, the corresponding swimming is not unlike *carangiform* swimming in which most of the body is rigid except the tail, which moves back and forth (Figure 5).

The two controllers that produce regular oscillations and reach the highest speeds (controllers 9 and 10), exhibit an anguiliform swimming, which is very similar to that of the real lamprey. Similarly to the lamprey, they have left and right neurons oscillating out of phase, and phase lags between segments that are almost constant over the spinal cord (although, for controller 10, the signal shapes slightly vary from head to tail), leading to caudally directed traveling waves (Figures 6 and 7).

Controller 9 has the peculiarity of including only one oscillator. It is also the controller with the smallest number of neurons, with one oscillatory circuit in the first segment (closest to the head) and two neurons per segment in the other segments (Figure 8). That rostral oscillator entrains the neurons in the other segments through a chain of caudally directed excitatory connections. Because of the synaptic time constants, these neurons oscillate with a lag compared to their rostral neighbors, which leads to the observed anguiliform swimming. The fact that only the first segment contains an oscillator is due to the border effect of the substrate. The same DRAW instruction is responsible both for the creation of a connection completing an oscillator circuit in the first segment, and for the creation of intersegmental links connecting each of the other segments to the preceding one (Figure 9). Before evaluating this controller,

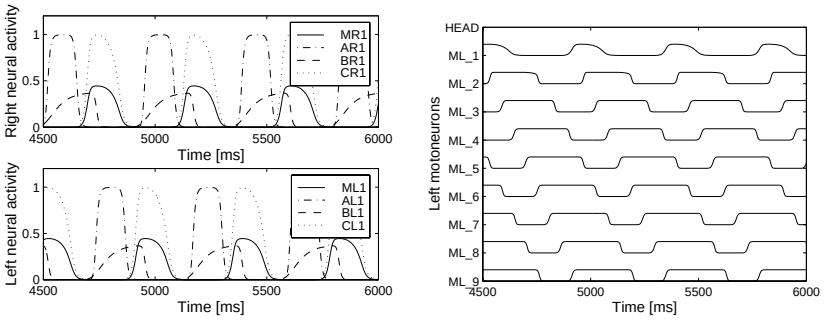


Figure 6. Neural activity in controller 9. Left: Neural activity of left and right neurons and the signals sent to the muscles in segment 1 (the oscillator segment, see text). Right: Signals sent to the left muscles along the nine segments.

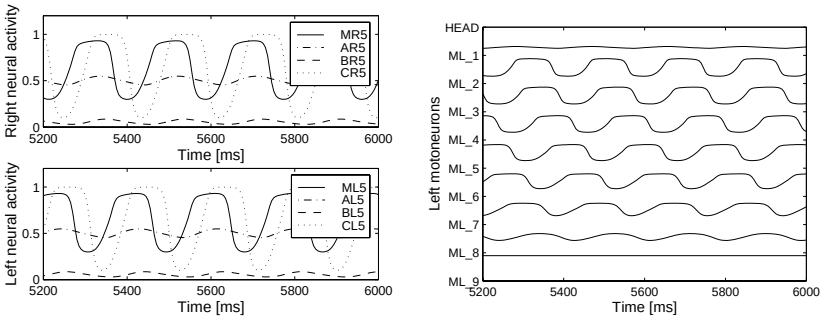
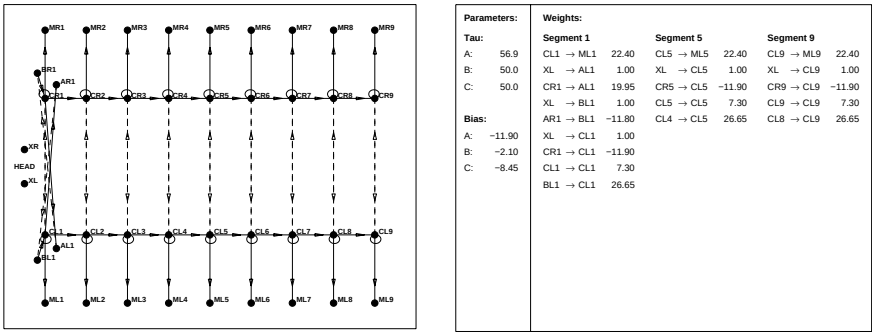


Figure 7. Neural activity in controller 10. Left: Neural activity of left and right neurons and the signals sent to the muscles in segment 5. Right: Signals sent to the left muscles along the nine segments.



Parameters:

Tau:

A:	56.9
B:	50.0
C:	50.0

Bias:

A:	−11.90
B:	−2.10
C:	−8.45

Weights:

Segment 1		Segment 5		Segment 9	
CL1 → ML1	22.40	CL5 → ML5	22.40	CL9 → ML9	22.40
XL → AL1	1.00	XL → CL5	1.00	XL → CL9	1.00
CR1 → AL1	19.95	CR5 → CL5	−11.90	CR9 → CL9	−11.90
XL → BL1	1.00	CL5 → CL5	7.30	CL9 → CL9	7.30
AR1 → BL1	−11.80	CL4 → CL5	26.65	CL8 → CL9	26.65
XL → CL1	1.00				
CR1 → CL1	−11.90				
CL1 → CL1	7.30				
BL1 → CL1	26.65				

Figure 8. Architecture of controller 9, one of the two controllers producing anguiliform swimming. Left: Neurons on the substrate. For reasons of clarity, the lateral axis of the substrate has been multiplied by ten. Ml_i and Mr_i represent the left and right muscle nodes of segment i, Xl and Xr are the input nodes. A, B, and C are three types of neurons. Excitatory and inhibitory connections are represented through continuous and dashed lines, respectively. Each neuron also receives a connection from the input nodes, which is not shown here. Right: Parameters and connection weights. Only the weights of the connections to the left neurons are given; the others are symmetric.

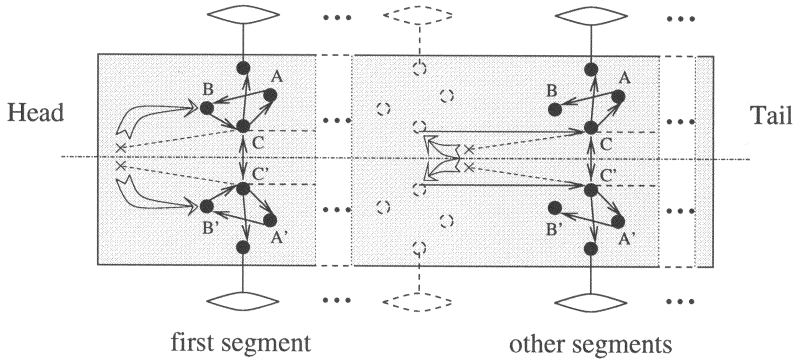
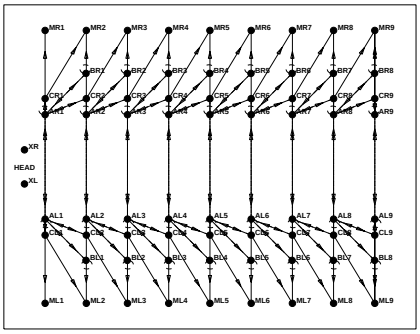


Figure 9. Side effect during the development of controller 9: During the development, a DRAW instruction was read by all cells C, and a connection was created toward each cell C from the cell closest to the instruction's target point (in the coordinates of the local frame attached to the cell C, and represented as a cross on the figure). In all but the first segment, this led to the creation of an intersegment connection with the homologous cell C in the preceding segment. In the first segment, however, as there was no preceding segment, the closest cell to the target point was cell B in the first segment itself. Therefore, a connection from cell B to cell C was created in the first segment. That connection complements a recurrent circuit implementing an oscillator whose periodic activity is propagated into the other segments of the animat by the intersegment connections. Note that for reasons of clarity, we have reversed left and right cells A compared to Figure 8.

the neurons labeled A and B in the figure are pruned in all but the first segment because they have no efferent connections toward the muscles or the C neurons, which implies that these cells cannot influence the behavior of the simulated lamprey and thus need not be simulated.

Controller 10 has a similar organization to controller 8, with one oscillator per segment being distributed over both sides of the spinal cord through inhibitory contralateral connections (Figure 10). The segments are interconnected over the spinal cord by closest neighbor coupling (because of the limitation of the length of a connection, parameter *r* of the GROW and DRAW instructions, no longer coupling was developed). Note that of the two controllers producing anguiliform swimming, controller 10 is more robust against "lesions" than controller 9 in the sense that destroying neurons or connections in the head oscillator or in the coupling between segments of the controller 9 has a significantly larger impact on the swimming pattern than for controller 10.

These four swimming controllers (controllers 7, 8, 9, and 10) modulate the speed of swimming when the level of tonic input is varied. They need the tonic input to produce



Parameters:

Tau:

A:	1600.0
B:	29.2
C:	16.5

Bias:

A:	−16.00
B:	−16.00
C:	2.00

Weights:

Segment 1		Segment 5		Segment 9	
AL1 → ML1	27.00	CL4 → ML5	31.80	CL8 → ML9	31.80
XL → AL1	28.45	AL5 → ML5	27.00	AL9 → ML9	27.00
BL1 → AL1	−6.10	XL → AL5	28.45	CL9 → ML9	31.80
CL2 → AL1	−24.75	BL5 → AL5	−6.10	XL → AL9	28.45
AL1 → AL1	−23.95	CL6 → AL5	−24.75	CL9 → AL9	−6.10
XL → BL1	1.00	AL5 → AL5	−23.95	CL9 → AL9	−24.75
BL1 → BL1	−16.20	XL → BL5	1.00	AL9 → AL9	−23.95
AL1 → BL1	24.70	BL5 → BL5	−16.20	XL → CL9	1.00
XL → CL1	1.00	AL5 → BL5	24.70	CR9 → CL9	−6.95
CR1 → CL1	−6.95	XL → CL5	1.00	AR9 → CL9	−27.90
AR1 → CL1	−27.90	CR5 → CL5	−6.95	AL8 → CL9	30.70
AL1 → CL1	30.70	AR5 → CL5	−27.90		
		AL4 → CL5	30.70		

Figure 10. Architecture of controller 10, one of the two controllers producing anguiliform swimming. Left: Neurons on the substrate. Right: Parameters and connection weights. For details, see Figure 8 caption.

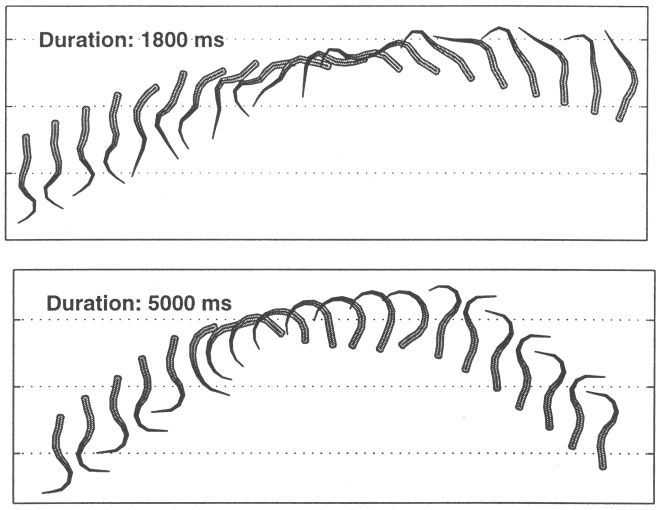


Figure 11. Top: slow turning induced by controller 10. Bottom: Sharp turning induced by controller 9.

oscillations; they converge to an equilibrium state, and hence a motionless body, when no input is given. Once sufficient input is applied, the speed of swimming increases with the level of input, and these controllers have a range of excitations in which the speed increase is monotonic with the excitation (even with smaller excitation steps than those used in the evaluation of the fitness value). Table 4 gives the speed ranges that can be obtained within that monotonic region. This monotonic relation is important, from a control point of view, to ensure that an increase of excitation amounts to an increase of speed.

Controllers 9 and 10 also offer good control of the direction of swimming. When, during swimming, an asymmetrical excitation is applied to these networks for a fixed duration, the lamprey turns proportionately to the asymmetry of excitation. The effect of the asymmetry is to change the duration of bursts between left and right muscles leading to a change of direction toward the side with the longest bursts. If the asymmetry is small (e.g., $\pm 10\%$ of the excitation level), the lamprey goes on propagating a traveling undulation and therefore swims in a circle until the asymmetry is stopped (Figure 11, top). If the asymmetry is large (for instance $\pm 30\%$ of the excitation level), the undulations almost stop as one side becomes completely contracted. This bending leads the lamprey to turn very sharply and allows important changes of direction when the duration of the asymmetry is short (Figure 11, bottom).

7 Discussion

7.1 Method

We have presented how an evolutionary algorithm with a developmental encoding scheme could be used to develop interesting swimming controllers. This demonstrated that the method has the potential to develop, given a fixed body structure, efficient neural controllers for animal-like locomotion, namely controllers that can transform simple commands concerning the speed and direction of motion into rhythmic signals sent to the multiple actuators.

7.1.1 Evolution and Search Space

The fact that the evolutionary process determines the number of neurons, the number of connections, and the parameters (synaptic weights, time constants, and biases) of the neurons leads to a search space with, in principle, an infinite number of dimensions, and we therefore canalized the search in several ways to make the task of the evolutionary algorithm easier.

First, we chose the initial positions and orientations of the precursor cells and output nodes in the substrate, which allowed us to force the developed controllers to be symmetrical and organized in segments, thus reducing the overall number of parameters to be evolved. While such symmetry hypotheses are commonly used when evolving locomotion controllers with fixed topologies [3, 28, 35], SGOCE allows evolution of *variable* symmetric topologies.

Second, the choice of a specific grammar made it possible to reduce further the size of the search space by considering only a subspace of the developmental program space. Moreover, the grammar indirectly limited the numbers of neurons in the evolved controllers, therefore restricting the search space to a finite number of dimensions. The limited numbers of neurons also helped to keep the evaluation time of the corresponding neural networks within reasonable bounds.

As mentioned, the evolved controllers depend on several geometrical parameters fixed by the experimenter, such as the number and the position of the precursor cells, the position of the muscle nodes, the maximum length of a connection, and so on, that have partly determined the architecture of the evolved solutions. An interesting extension of this work would be to analyze the effect of these parameters, and potentially having them evolving as well.

7.1.2 Evolutionary Algorithms for Neural Networks

From a general point of view, using evolutionary algorithms is an interesting technique for designing neural networks compared to traditional learning algorithms (such as variations of the backpropagation algorithm for dynamical networks; see, e.g., [36]) because of their flexibility. The fitness function of an evolutionary algorithm does not, for instance, need to be differentiable or even continuous like error functions minimized by most learning algorithms. There is also no need to provide a specific oscillation (limit cycle) that the network should learn. This has allowed us to characterize the desired behavior of the controller at a higher level, by determining desired characteristics of the swimming control in the mechanical simulation. These features make evolutionary algorithms a popular design technique in the field of neural networks applied to the control of animats (see [3, 9, 22, 31, 35] for evolutions of locomotion controllers and [15, 16] for a review of evolutions of behavior controllers).

7.1.3 Comparison with Previous Evolutions of Swimming Controllers for the Lamprey

In [26], we used a staged evolution approach to generate swimming controllers, with a first stage in which we evolved segmental oscillators and a second stage in which we evolved the coupling between the oscillators. We used the same neuron models as Ekeberg [11], which have three state variables instead of one. For the first stage, the fitness function was based on neural aspects, such as the range of frequencies of oscillation, and in the second stage, the fitness function was based on both neural aspects and mechanical aspects, such as the speed of swimming. We used a direct encoding of the synaptic weights in the first stage, and of the extent of the interconnections in the second stage.

The experiment reported here concentrates on producing control mechanisms for animal-like locomotion rather than visiting the space of potential neural configurations

around the biological solution. In contrast to [26], we use simpler neurons and a more compact encoding—the developmental encoding SGOCE—and we evolve swimming controllers in a single stage with a fitness function only based on observable characteristics of the mechanical simulation. This new approach brings several interesting features.

First, an interesting aspect of the developmental encoding is the fact that the number of neurons and of connections is not fixed a priori but is determined by the evolutionary process. In [26], we evolved the number of neurons and connections indirectly by fixing a maximum number of neurons and evolving the weights of all possible connections while using a random pruning operator that little by little led to circuits with fewer connections and potentially fewer neurons.

Second, another interesting aspect of the developmental encoding is that the developmental instructions can participate in the construction of different neural mechanisms depending on the local context, as illustrated in the development of controller 9, when the same DRAW instruction was used either to complete an oscillatory circuit in the first segment or create intersegmental connections in the other segments. Such a property is due to the side effects that occur at the extremities of the segment chain and to the fact that the instructions for developing connections are context-dependent. The evolutionary algorithm was opportunistic enough to take advantage of that property. One idea to make such opportunistic solutions appear more frequently would be to define more context-dependent developmental mechanisms (e.g., by having values of synaptic weights that depend on the position of the neuron).

Finally, swimming controllers have been developed in one stage only, with a fitness function that only specifies desired characteristics for the mechanical simulation, without considering neural aspects. Although the staged approach of the previous work allowed swimming controllers to be generated in less computational time, having a fitness evaluation only based on mechanical aspects is necessary for control problems where neural requirements are not known and where only the desired behavior of the mechanical system can be characterized. An interesting application of the technique could be the generation of a locomotion controller for a swimming or walking robot, for instance. An external camera filming the locomotion gait could then be used for measuring the characteristics taken into account by the fitness function (see below).

7.1.4 Comparison with Previous Applications of SGOCE

In [31] walking controllers for a six-legged animat were evolved and several similarities can be noticed between the corresponding experiments and the ones described here. First, the same model of neurons and a similar model of muscles were used in both cases. Moreover, a left-right symmetry and a segmental organization of the body and nervous system were hypothesized. Second, the set of developmental instructions and the syntactical constraints used were nearly identical, the only difference being the addition of the instruction SETINPUTW in the current work. Furthermore, the same parameter values were used by the evolutionary algorithm for solving both tasks, the only difference to [31] being that in the work reported here the population size was reduced because of the longer time required by the evaluations. Finally, in both cases, locomotion controllers involving central pattern generators were successfully evolved.

The main differences between the experiments described here and those of [31] lie in the evaluation strategies and in the inclusion of command cells. While the six-legged animat experiments used a global fitness criterion involving measures of body speed and leg movements, the current work uses a more elaborate fitness function that combines four criteria and calls for several evaluations of a given individual with different command settings.

Also, command cells were explicitly included in the current work, which allowed neural circuitries to be developed whose swimming patterns could be initiated and

modulated by simple input signals. Note that while no command input was used in the first stage of the evolution of six-legged locomotion controllers—which led to the evolution of constant-speed, straight-walking controllers—those controllers were, however, found to be able to stop [31] or to turn [32] when connected to a second, higher-level neural network. More experiments are needed to determine to what extent command inputs are useful or necessary for solving tasks that involve controlling the speed and direction of locomotion. Preliminary arguments in favor of the inclusion of command cells (instead of first evolving a neural layer for pattern generation, followed by a layer for the control of speed and direction) include the generation of more compact neural networks and an optimized controllability. In [14], where SGOCE is applied to the control of a real hexapod robot, an incremental strategy involving such command inputs is proposed.

7.2 Evolved CPGs

In the work described here, a variety of different controllers have been evolved. The controllers differ both in their behaviors and their configurations. The evolved controllers present three kinds of neural activity: nonoscillating signals that propel the lamprey forward through an initial thrust, chaotic oscillations, and stable oscillations. The corresponding configurations differ in the number of neurons, their parameters, and their connectivity. In particular the solutions can be classified depending on their numbers of oscillators: no oscillators, only one oscillator, and one or more oscillators per segment.

7.2.1 Evolved Swimming Patterns

The controllers without oscillators, which propel the body forward by an initial thrust, are not very interesting from a control point of view as they cannot sustain the speed of motion. These controllers have taken advantage of the limited duration of the evaluations. To prevent the emergence of this kind of solution, further evolutions should be carried out with longer simulations that would significantly reduce their fitness value. The controllers producing unstable oscillations are also of limited interest because their speed and direction of swimming are not constant. These controllers would therefore be difficult to use by higher control centers. A simple way to prevent these solutions from being generated could be to add a factor in the fitness function rewarding regularity of both the speed and the direction.

Four controllers have been developed that can be described as central pattern generators. They generate stable and permanent oscillations when receiving a tonic (i.e., nonoscillating) input. Of these controllers, the two that reach the highest speeds produce an anguiliform swimming very similar to that of the lamprey. Segments oscillate with left and right neurons out of phase, and there is a constant phase lag between segments that leads to a wave of neural activity traveling from head to tail. These two controllers offer good control of the speed and the direction. Increasing the amplitude of both tonic inputs amounts to increasing the speed of swimming. One of our requirements was that this increase be monotonic, to facilitate control and to be certain that an increase of excitation amounts to an increase of speed. We can therefore define for both controllers a range of excitations in which that condition is respected. When asymmetrical levels of inputs are applied, turning is induced because of a difference in the duration of the muscular bursts between both sides. Although we have not required a monotonic relation between the turning capacity and the asymmetry of excitation (turning was tested at only one asymmetry), the turning angle increases with the amount of asymmetry. For both controllers, two kinds of turning can be obtained: A small asymmetry leads to slow turning in which the undulations continue to travel, and a large asymmetry leads to sharp turning with a strong bending of the body. Al-

though this means that the body loses speed, it allows turning on the spot. For both types of turn, straight swimming resumes when the asymmetry ceases.

7.2.2 Comparison of the Best Controllers with the Lamprey's CPG

Controller 9 has a very simple neural configuration for creating the traveling wave of neural activity. It is made of an oscillator in the first segment (the segment closest to the head) that entrains, through closest neighbor excitatory connections, a chain of single neurons on both sides of the spinal cord. Controller 10 has a more complex configuration with oscillators in each segment. This controller is closer to the biological connectivity found in the lamprey [6, 19, 20, 21]. The spinal cord of the lamprey is indeed considered to be an interconnection of segmental oscillators, as small parts of the spinal cord up to 2 segments can be made to oscillate in isolation (the whole spinal cord is made of approximately 100 segments). Both evolved controllers share the property of the lamprey that the frequency of oscillation increases with the level of excitation. In the lamprey, this has been shown *in vitro* by measuring the frequency of oscillation in excitatory baths with different concentrations. Another interesting property of the lamprey is that the phase lag between segments relative to the cycle duration remains constant for different frequencies of oscillation. This means that the lamprey maintains a constant wavelength of the undulation (corresponding to approximately the length of the body) for any frequency of oscillation. In controller 9 this is not the case, as the relative lag between segments changes significantly with the frequency; in controller 10, however, the change of relative phase lag changes only slightly for important changes of the frequency (data not shown, see [27]). Controller 10 therefore shares the most properties with the biological CPG of the lamprey.

By contrast with our previous work in which we developed swimming controllers with some predefined similarities with the biological configuration and with some requirements on the neural activity, we generated here controllers with only a characterization of the desired behavior in terms of the control of the speed and the direction of swimming. It is therefore interesting to see that the combination of evolution, development, and dynamical neurons has led to best controllers sharing several similarities with the lamprey, the main one being the anguiliform swimming. Further experiments would be needed to determine how much these similarities are due to mechanical considerations (anguiliform swimming being optimal for our fitness function given the lamprey body and the mechanical simulation) or to neuronal considerations (producing a traveling wave with constant wavelength along the spinal cord is the best solution available in the space of potential neural configurations), or both. In any case, the similarities suggest that the fitness function, the mechanical simulation, and the space of potential neural configurations have determined constraints for the artificial evolution that are not too different from those that have directed the natural evolution of the swimming circuitry of the real lamprey.[5]

7.3 Further Directions

The evolved controllers work in an open loop manner, in the sense that the mechanical simulation does not send feedback to the controller. In the lamprey, sensory feedback is provided by stretch sensitive cells, the edge cells, which are located on both sides of the spinal cord [44]. These cells are useful for coordinating the neural activity with the actual body state when swimming in unstationary water. Ekeberg has shown in particular that these cells are necessary for crossing a local speed barrier [12]. Similarly to

5 Note that our experiment is not intended to simulate natural evolution. A major difference with natural evolution, is, of course, that in natural evolution, body structure and control mechanisms are coevolved. Furthermore, the "fitness function" of natural evolution—the capacity to reproduce—creates significantly more constraints on the evolution of control mechanisms than our simple fitness function.

[26], where we evolved such a feedback, further experiments using the developmental encoding should include these cells to close the control loop.

As the speed and the direction of swimming can be modulated through two input signals, the movements of the simulated lamprey (in a plane) can be determined. The evolved controllers can therefore be used by higher neural centers for the generation of simple behaviors, with the two input nodes of the CPG providing the interface between the CPG and the higher control centers. A natural continuation of this work would therefore consist in extending the simulation by adding a simple simulated visual system to it, and in evolving higher neural centers for simple sensorimotor coordination experiments such as avoiding obstacles or approaching and attaching to a host (the lamprey is a parasite). In that respect, a long-term goal would be to reach the behavioral capacities shown by the artificial fishes [40, 41], but with a neural-based control mechanism. The aim of such a study would be to investigate how the evolved CPGs react to continuously changing commands, and what type of neural architectures are capable of making the transformation of sensory inputs into motor commands for simple coordination behaviors.

In another, more artificial-life-like, direction, an extension would be to investigate the dynamics of coevolution of body structures and control mechanisms. As mentioned earlier, the whole body structure as well as several characteristics of the neural controller were fixed by the experimenter, and several ways could be imagined to have them evolve simultaneously with the neural controllers. We would then come close to Sims' [38] and Ventrella's [43] work, but with more realistic, or at least closer to biological life, body structures and neural control mechanisms. Such an investigation may therefore give some insights on the evolutionary dynamics of natural evolution.

Finally, an ultimate test for the method would be to evolve CPGs for controlling the swimming of a real robot. This would require first developing controllers for a three-dimensional simulation (see, e.g., [12]) which, depending on the degree of freedom of the spinal joints, implies more muscles for controlling the pitch and roll movements and some feedback from vestibular systems to achieve roll stability. Second, the controllers would have to be transferred to the real robot with, potentially, *online* evolution to further adapt the controllers to the robot. The motivations for a swimming robot are numerous, the main ones being to obtain locomotion that is significantly more agile and energetically efficient than propeller propulsion (see, e.g., the ongoing project to build a lamprey robot [29] at the Marine Science Center, Northeastern University, and the Robotuna and Robopike projects at MIT's Department of Ocean Engineering). Concerning the evolutionary approach we propose, the motivation would be to test whether the method can be applied to a significantly more complex control problem than the two-dimensional swimming treated here.

8 Conclusion

This article has presented how swimming controllers for a mechanical simulation of a lamprey can be designed using an evolutionary algorithm with a developmental encoding scheme. Developmental programs determine how neurons on a two-dimensional substrate divide and get connected to each other. Swimming controllers are developed given the fixed body structure of the lamprey and given a high-level characterization of the desired behavior in terms of the control of the speed and the direction of swimming.

Similarly to central pattern generators found in animals, neural circuitries are evolved that can produce and modulate relatively complex patterns of oscillations when receiving simple tonic input. By simply varying the amplitude of two input signals, the speed and the direction of swimming can be varied. The two best controllers produce an

anguiliform swimming very similar to that of the lamprey, with a traveling wave of neural activity propagating from head to tail.

Acknowledgments

We would like to thank John Hallam and Jean-Arcady Meyer for their comments on earlier versions of this paper. The facilities were provided by the University of Edinburgh and by the AnimatLab. Auke Jan Ijspeert is supported by grants from the Swiss National Research Fund and from the European Commission (Marie Curie Fellowship).

References

1. Beer, R. D. (1990). *Intelligence as adaptive behavior, an experiment in computational neuroethology*. San Diego: Academic Press.
2. Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, *3*, 469–510.
3. Beer, R. D., & Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, *1*, 91–122.
4. Brooks, R. A. (1989). A robot that walks: Emergent behavior from a carefully evolved network. *Neural Computation*, *1*, 253–262.
5. Buchanan, J. T. (1992). Neural network simulations of coupled locomotor oscillators in the lamprey spinal cord. *Biological Cybernetics*, *66*, 367–374.
6. Buchanan, J. T., & Grillner, S. (1987). Newly identified "glutamate interneurons" and their role in locomotion in the lamprey spinal cord. *Science*, *236*, 312–314.
7. Cliff, D. (1995). Computational neuroethology. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 626–630). Cambridge, MA: MIT Press.
8. Cruse, H., Brunn, D. E., Bartling, Ch., Dean, J., Dreifert, M., Kindermann, T., & Schmitz, J. (1995). Walking: A complex behavior controlled by simple networks. *Adaptive Behavior*, *3*, 385–418.
9. de Garis, H. (1990). Genetic programming: Building artificial nervous systems using genetically programmed neural network modules. In B. W. Porter & R. J. Mooney (Eds.), *Proceedings of the Seventh International Conference on Machine Learning* (pp. 132–139). Palo Alto, CA: Morgan Kaufmann.
10. Delcomyn, F. (1980). Neural basis for rhythmic behaviour in animals. *Science*, *210*, 492–498.
11. Ekeberg, Ö. (1993). A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics*, 69, 363–374.
12. Ekeberg, Ö., Lansner, A., & Grillner, S. (1995). The neural control of fish swimming studied through numerical simulations. *Adaptive Behavior*, *3*, 363–384.
13. Ekeberg, Ö., Wallén, P., Brodin, L., & Lansner, A. (1991). A computer-based model for realistic simulations of neural networks I: The single neuron and synaptic interaction. *Biological Cybernetics*, *65*, 81–90.
14. Filliat, D. (1998). *Evolution de réseaux de neurones pour le contrôle d'un robot hexapode*. (Tech. Rep.). Paris: AnimatLab, ENS.
15. Floreano, D. (1997). Evolutionary mobile robotics. In D. Quagliarelli, J. Periaux, C. Poloni, & G. Winter (Eds.), *Genetic algorithms in engineering and computer science*. Chichester, UK: Wiley. 341–366.
16. Floreano, D. (1998). Evolutionary robotics in artificial life and behavior engineering. In T. Gomi (Ed.), *Evolutionary robotics*. Ontario: AAI Books. 77–103.
17. Getting, P. A. (1988). Comparative analysis of invertebrate central pattern generators. In A. H. Cohen, S. Rossignol, & S. Grillner (Eds.), *Neural control of rhythmic movements in vertebrates* (pp. 101–127). New York: Wiley.

18. Grillner, S. (1996, January). Neural networks for vertebrate locomotion. *Scientific American*, 48–53.

19. Grillner, S., Buchanan, J. T., Wallén, P., & Brodin, L. (1988). Neural control of locomotion in lower vertebrates. In A. H. Cohen, S. Rossignol, & S. Grillner (Eds.), *Neural control of rhythmic movements in vertebrates* (pp. 1–40). New York: Wiley.

20. Grillner, S., Degliana, T., Ekeberg, Ö., El Marina, A., Lansner, A., Orlovsky, G. N., & Wallén, P. (1995). Neural networks that co-ordinate locomotion and body orientation in lamprey. *Trends in Neuroscience*, *18*(6), 270–279.

21. Grillner, S., Wallén, P., & Brodin, L. (1991). Neuronal network generating locomotor behavior in lamprey: Circuitry, transmitters, membrane properties, and simulation. *Annual Review of Neuroscience*, *14*, 169–199.

22. Gruau, F. (1995). Automatic definition of modular neural networks. *Adaptive Behavior*, *3*, 151–184.

23. Hopfield, J. J. (1984). Neurons with graded response properties have collective computational properties like those of two-state neurons. In *Proceedings of the National Academy of Sciences* (Vol. 81, pp. 3088–3092). Washington, DC: The National Academy of Sciences.

24. Ijspeert, A. J., Hallam, J., & Willshaw, D. (1998). *Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology*. (Research Paper No. 876). University of Edinburgh Department of Artificial Intelligence.

25. Ijspeert, A. J., Hallam, J., & Willshaw, D. (1998). From lampreys to salamanders: Evolving neural controllers for swimming and walking. In R. Pfeifer, B. Blumberg, J.-A. Meyer, & S. W. Wilson (Eds.), *From Animals to Animats: Proceedings of the Fifth International Conference of the Society for Adaptive Behavior* (pp. 390–399). Cambridge, MA: MIT Press.

26. Ijspeert, A. J., Hallam, J., & Willshaw, D. (in press). Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Adaptive Behavior*, 7(2).

27. Ijspeert, A. J., & Kodjabachian, J. (1998). *Evolution and development of a central pattern generator for the swimming of a lamprey*. (Research Paper No. 926). University of Edinburgh, Department of Artificial Intelligence.

28. Jakobi, N. (1998). Running across the reality gap: Octopod locomotion evolved in a minimal simulation. In P. Husbands & J.-A. Meyer (Eds.), *EvoRobot'98: Proceedings of the First European Workshop on Evolutionary Robotics* (pp. 39–58). Berlin: Springer-Verlag.

29. Jalbert, J., Kashin, S., & Ayers, J. (1995). A biologically-based undulatory lamprey-like auv. In *Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium* (pp. 39–52). Monterey, CA: Naval Postgraduate School.

30. Kodjabachian, J., & Meyer, J.-A. (1995). Evolution and development of control architectures in animats. *Robotics and Autonomous Systems*, *16*, 161–182.

31. Kodjabachian, J., & Meyer, J.-A. (1998). Evolution and development of modular control architectures for 1-d locomotion in six-legged animats. *Connection Science, 10*, 211–237.

32. Kodjabachian, J., & Meyer, J.-A. (1998). Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects. *IEEE Transactions on Neural Networks, 9*(5), 796–812.

33. Kopell, N. (1995). Chains of coupled oscillators. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 178–183). Cambridge, MA: MIT Press.

34. Lewis, M. A. (1996). *Self-organization of locomotory controllers in robots and animals*. Unpublished doctoral dissertation, University of Southern California.

35. Lewis, M. A., Fagg, A. H., & Bekey, G. A. (1993). Genetic algorithms for gait synthesis in a hexapod robot. In Y. F. Zheng (Ed.), *Recent trends in mobile robots*. River Edge, NJ: World Scientific.

36. Pearlmutter, B. A. (1995). Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, *6*(5), 1212–1228.

37. Schaffer, J., Whitley, D., & Eskelman, L. J. (1992). Combinations of genetic algorithms and neural networks: A survey of the state of the art. In J. Schaffer & D. Whitley (Eds.), *Combinations of genetic algorithms and neural networks* (pp. 1–37). Los Alamitos, CA: IEEE Computer Society Press.

38. Sims, K. (1994). Evolving virtual creatures. In *Computer Graphics Proceedings, Annual Conferences Series* (pp. 15–24). New York: Association for Computing Machinery.

39. Spencer, G. (1994). Automatic generation of programs for crawling and walking. In K. E. Kinnear (Ed.), *Advances in genetic programming* (pp. 335–353). Cambridge, MA: MIT Press.

40. Terzopoulos, D., Rabie, T., & Grzeszczuk, R. (1996). Perception and learning in artificial animals. In C. G. Langton and T. Shimohara (Eds.), *Proceedings, Artificial Life V* (pp. 354–361). Cambridge, MA: MIT Press.

41. Terzopoulos, D., Tu, X., & Grzeszczuk, R. (1994). Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, *1*, 327–351.

42. Usami, Y., Saburo, H., Inaba, S., & Kitaoka, M. (1998). Reconstruction of extinct animals in the computer. In C. Adami, R. Belew, H. Kitano, & C. Taylor (Eds.), *Proceedings of the Sixth International Conference on Artificial Life* (pp. 173–177). Cambridge, MA: MIT Press.

43. Ventrella, J. (1998). Attractiveness vs. efficiency. In C. Adami, R. K. Belew, H. Kitano, & C. E. Taylor (Eds.), *Proceedings of the Sixth International Conference on Artificial Life* (pp. 178–186). Cambridge, MA: MIT Press.

44. Viana Di Prisco, G., Wallén, P., & Grillner, S. (1990). Synaptic effects of intraspinal stretch receptor neurons mediating movement-related feedback during locomotion. *Brain Research*, *530*, 161–166.

45. Wallén, P., Ekeberg, Ö., Lansner, A., Brodin, L., Traven, H., & Grillner, S. (1992). A computer-based model for realistic simulations of neural networks II: The segmental network generating locomotor rhythmicity in the lamprey. *Journal of Neurophysiology*, *68*, 1939–1950.

46. Williams, T. L. (1992). Phase coupling by synaptic spread in chains of coupled neuronal oscillators. *Science*, *258*, 662–665.

47. Williams, T. L., & Sigvardt, K. A. (1995). Spinal cord of lamprey: Generation of locomotor patterns. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 918–921). Cambridge, MA: MIT Press.

48. Yao, X. (1993). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, *8*(4), 539–567.