Evolution of Bilateral Symmetry in Agents Controlled by Spiking Neural Networks

Nicolas Oros, Volker Steuber, Neil Davey, Lola Cañamero, and Rod Adams

Science and Technology Research Institute, University of Hertfordshire, AL10 9AB, United Kingdom

{N.Oros, V.Steuber, N.Davey, L.Canamero, R.G.Adams}@herts.ac.uk

***Abstract*—We present in this paper three novel developmental models allowing information to be encoded in space and time, using spiking neurons placed on a 2D substrate. In two of these models, we introduce neural development that can use bilateral symmetry. We show that these models can create neural controllers for agents evolved to perform chemotaxis. Neural bilateral symmetry can be evolved and be beneficial for an agent. This work is the first, as far as we know, to present developmental models where spiking neurons are generated in space and where bilateral symmetry can be evolved and proved to be beneficial in this context.**

I. Introduction

In order to investigate the importance of bilateral symmetry in artificial neural development, here we introduce three different novel models of a developmental program that grow spiking neural networks on a two-dimensional substrate. Each of these models has different degrees of allowed or enforced symmetry. These developmental programs are evolved, using a genetic algorithm, to allow simulated agents to perform chemotaxis. This paper begins with a basic introduction on symmetry in nature and how it has been modeled in artificial evolutionary models. Then, we introduce our developmental model, the agent used and the task it had to perform. Further, we describe in more detail our three different models. Then, the simulation and genetic algorithm parameters are presented. This section is followed by the results, the discussion and finally the conclusion.

A. Symmetry

For centuries, people have observed and been fascinated by symmetrical patterns found in nature [1-3]. In our minds, symmetry is often related to something beautiful, well balanced or well proportionate [3]. It has been shown that in many species (even in humans), female prefer males that have symmetrical displays [4]. One possible reason to explain this phenomenon is that symmetry might reflect the high quality of a signaler. Another reason could be that individuals have evolved recognition systems that have common properties and are capable of generalization, and from this could emerge a high sensitivity to symmetries [4]. In living organisms, symmetries arise as a side effect of the creation of axes that will guide cells during development [1-3, 5-8]. Cells divide and migrate following gradients that form these axes. They might also create or modify gradients and rearrange themselves to form the most thermodynamically stable pattern [6]. Therefore, it is very likely that cells will be placed symmetrically along different axes to have a system in a state of equilibrium [3]. But due to developmental noise, even the most bilaterally symmetrical animals do not show perfect symmetry. Also, many vertebrates are mainly bilaterally symmetrical about the midline of the body but they have many internal organs that are not bilaterally symmetrical (for example in humans: heart, stomach, spleen…) [5-8]. Even if the emergence of a bilateral body plan was a key step in evolution, new axes were defined that differentiated head and foot, back and front and left from right, and allowed asymmetrical parts to be created and eventually lead to more complex organisms.

B. Evolutionary Computation

In order to understand the importance of symmetry in development, certain researchers in artificial intelligence have created abstract developmental models that generate neural controllers for robots or simulated agents. It is always a difficult task to create robust and adaptable neural controllers for agents that can perform many different actions. It is even more difficult if you want to reuse existing controllers and add new modules so an agent can learn and perform new tasks. A promising trend is to evolve neural networks using evolutionary computation. There are different approaches in this research area and many different ways to encode evolving features into genes [9-14]. A certain amount of work has been done in evolutionary computation on encoding spatial neural networks [15], with symmetrical structure using L-systems [16-21] and grammatical encoding [22, 23]. Stanley also created abstract models generating representations of symmetrical patterns [24, 25]. To the best of our knowledge, no one has created a developmental model generating spiking neural controllers placed in 2D spaces, where bilateral symmetry can be evolved, and improved the performance of an agent to perform certain tasks.

C. Our Approach

In this study, we used developmental programs that allowed information to be encoded as spatio-temporal neural activity patterns. We created three new developmental models initially inspired by Kodjabachian and Meyer's

SGOCE paradigm [26-30] and NEAT [24, 25, 31]. By using them, we wanted to see how bilateral symmetry in neural networks could be generated and affect the behavior of a simulated robot. In our models, a developmental program was expressed in a genome and when executed, it would create one or more intermediate neurons with one or more connections to make the whole neural network grow. Like in [31, 32], one of the key ideas in our approach is based on complexification. An initial genome is first composed of only one gene creating only one neuron when expressed. Then, during evolution, new genes can be added via mutations creating more neurons and more connections, therefore adding more complexity to the system. Another important concept of this model is targeting [32]. We used a 2D neural substrate where spiking neurons (with synaptic integration and conduction delays) are placed and can grow connections to target locations. Evolution can therefore generate neural networks able to encode external information as spatio-temporal patterns.

We first created a model where parameters of each neuron were encoded in the genome (NO_SYM). We then created two variations of it allowing bilateral symmetrical clones of neurons to be created. The first one allowed the evolution of symmetrical neurons (EVO_SYM) and the second one enforced the symmetry for every neuron (ENF_SYM). We also decided to have neural development performed in two stages: first creating every neuron on the substrate, then creating all the connections. This was inspired by biological systems where neurons first divide, then migrate to a certain location and finally create connections [5, 6, 33]. Some neurons might eventually die but we decided not to model apoptosis in our model to deal with complexity incrementally.

D. The agent and its Task

We decided to evolve an agent to perform a simple task which was to stay inside a chemical concentration in a simulated continuous environment (Fig. 3). The agent has two wheels, one on each side of the agent, providing a differential steering system. Each wheel is controlled by two motor neurons providing forward and backward propulsion. The agent also has two antennae placed on the front of the agent, one orientated on the left and the other one on the right. Each antenna is linked to a sensory neuron. The two antennae are separated widely enough to detect the presence of the chemical gradient (Fig. 1). To control the agent, we used a spiking neural network. The sensory and motor neurons placed on the neural substrate form the initial neural network (Fig. 2).The complete neural network was created by using a developmental program.

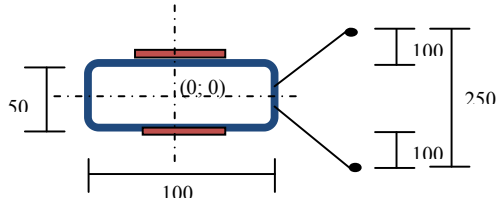

Fig. 1. Properties of an agent equipped with two wheels and two antennae. Units are arbitrary.

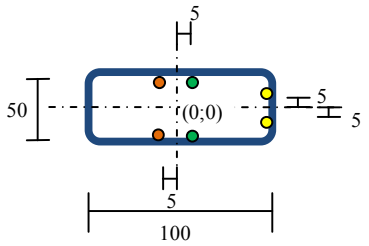

Fig. 2. 2D substrate of an agent with initial neural network. The two sensory neurons are shown on the right in yellow. The motor neurons move the agent forward (green) or backward (orange) by turning the wheel.

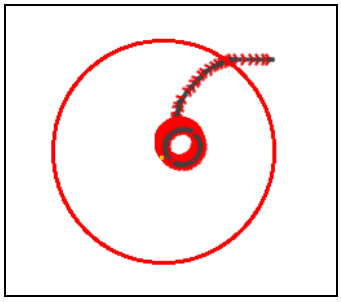

Fig. 3. Path of an agent moving towards the middle of a fixed chemical concentration (red circle). The concentration is a linear gradient where the maximum value is situated in the middle.

II. Methods

A. Spiking Neurons

We used a leaky-integrate and fire model with synaptic integration and conduction delays already described in [34, 35]. We also used a realistic model of noise in the form of an diffusive OU (Ornstein-Uhlenbeck) current [36]. This form of colored noise reproduces the subthreshold voltage fluctuations in real neuronal membranes. We added this noise to the total input current of each neuron. The noise current $I(t)$ is described by:

$$\frac{dI(t)}{dt} = -\frac{1}{\tau_I}\left(I(t) - I_0\right) + \sqrt{D}\,\xi(t) \qquad (1)$$

where τ_I denotes the current noise time constant (2ms in our case), I_0 is the mean noise current (0 in our case), $D = 2\sigma^2 / \tau_I$ is the noise diffusion coefficient, σ is the standard deviation (0.0007 in our case) and $\xi(t)$ is a Gaussian white noise (with mean = 0 and standard deviation = 1). The motor neurons used to control the wheels are modeled in the same way. However, the sensory neurons are based on this model but have a different expression to calculate the input current. We created a model of a spiking sensory neuron in which the chemical concentration is processed so that a quasi-linear relationship between the concentration and the firing rate of the sensor is produced. The sensory neurons were already described in detail in [34, 35, 37].

B. Chemical Concentration

We decided to use a simple model of chemicals that are not diffused and evaporated. The concentration is a linear

gradient where the maximum value is situated in the middle of the circular chemical concentration.

C. Agent Movements

In order to move the agent, we calculated the velocity *V(t)* (arbitrary unit) of each wheel using the following equation:

$$\tau_{motor}\frac{dV}{dt} = (V_0 - V) + K_v\left(\delta(t - t_f) - \delta(t - t_b)\right) \quad (2)$$

Where δ is the Dirac function (pulse) defined by $\delta(x) = 0$ when $x \neq 0$ and $\delta(x) = 1$ when $x = 0$.

We decided for simplicity that an agent should always move forward in the absence of any external input so we set up the parameters accordingly: $V_0 = 0.5$ is the default velocity (the agent is always moving straight by default), $K_v = 5$ is the speed coefficient, $\tau_{motor} = 0.05$ is the time constant in seconds, t_f is the time when the most recent spike was emitted by the motor neuron responsible to turn the wheel forward, t_b is the time when the most recent spike was emitted by the motor neuron responsible to turn the wheel backward. The agent was moved by calculating the velocity every time step.

III. Developmental Programs

A. Without Symmetry: NO_SYM

The developmental program constructing the neural network consists of a genome which is an array of modules. A module must have a gene, which we denote *N*, encoding the position (*x, y*) of an intermediate neuron, and can have genes encoding possible connections, denoted *C*. The neuron is placed on a 2D Cartesian coordinates system with its origin situated in the centre of the agent (Fig. 1 and 2). If a new module is created, it will be added to the end of the genome. A module is valid if it is composed of only one *N* gene but not if it is only composed of *C* genes. A *C* gene encodes the different parameters for a connection of a neuron. That includes an angle θ and a distance *d* to determine where it connects (see Section III. B), a synaptic strength (*w*) and a *type* (afferent or efferent). A neuron can also have connections even if they are not encoded in the module defining its properties. The reason is that other neurons can create efferent and afferent connections to this neuron.

When an agent is created, it only has an initial neural network (Fig. 2). There are no intermediate neurons, only motor neurons and sensors. If the genome of an agent is composed of at least one module, the complete neural network can be created by executing the developmental program expressed in the genes, reading the genome from the beginning to the end. With only one module, only one intermediate neuron will be created but it can have more than one connection. The neural network is constructed by the developmental program in two steps by reading the genome twice. First, all the neurons are created in the 2D substrate by reading all the *N* genes. Secondly, all the connections are created by reading all the *C* genes.

When reading a *C* gene, a target position for a given neuron is defined to determine to which neuron it will be connected to. The target position is given by the angle θ (in radians) and the distance parameter *d* relative to the neuron. A neuron creates a connection to the closest cell to this target position (Fig. 4). Self connections are therefore possible. Motor neurons cannot have output connections and sensory neurons cannot have input connections. A target position can be situated outside the substrate. In this case, a connection will still be created linking the closest neuron.

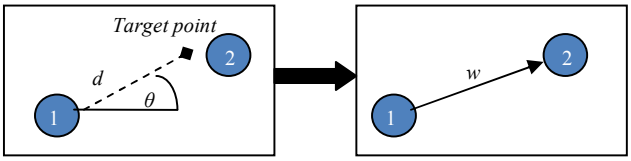

Fig. 4. Creation of a connection by neuron 1 in two steps. First, neuron 1 places a target point on the substrate depending on the distance *d* and angle θ parameters. Secondly, the closest neuron to the target point gets connected to neuron 1. The type of connection (input or output) depends on the parameter *type* and the synaptic strength (weight) is encoded by the parameter *w*.

TABLE 1
RANGES OF VALUES USED FOR THE PARAMETERS OF THE GENES

Parameters	Ranges of values
x	[-50,50]
y	[-25,25]
w	[-15,15]
θ	[0,2π]
d	[1,100]
type	afferent / efferent

B. Evolvable Symmetry: EVO_SYM

This model is a modification of NO_SYM. The main concept of this model is to introduce genetically encoded bilateral symmetry with respect to the longitudinal axis of the agent. The idea is that instead of encoding two neurons that are similar but are positioned on opposite sides of the midline (*x-axis*), the genome could encode only one neuron but with an extra evolvable parameter allowing the creation of its symmetrical clone; this allows compressing genetic information. In fact, the initial neural network is symmetrical, and therefore the evolutionary process should be able to use this important embedded feature. This model is based on an abstraction of a gradient that could form the horizontal axis. Compared to NO_SYM, *C* genes are still the same but *N* genes have an additional Boolean parameter *sym*. This parameter *sym* plays an important role. If it is activated (set to true), a clone of the actual neuron will be created and placed symmetrically to the *x-axis* (Figs. 5 & 6). If the parent neuron is situated on the *x-axis*, its clone will be created in a close random place around it.

The development of the neural network is very similar to NO_SYM. The only difference is that during the first step of development (creation of neurons), each created neuron will have a symmetrical clone if its parameter sym is set to true.

A clone of a neuron has its y parameter set to - y and all the connection parameters θ set to - θ. Therefore, the clone of a neuron is horizontally symmetric and its connections are also symmetric (Figs. 5 & 6). The neural growth is still performed in two steps by reading the genome twice. First, all the neurons (and their possible symmetrical clones) are created in the 2D substrate by reading all the N genes. Secondly, all the connections are created by reading all the C genes.

C. Enforced Symmetry: ENF_SYM

This developmental model is almost the same as NO_SYM. The only difference is the systematic creation of a symmetrical clone for every neuron. Every time a neuron is added to the substrate by executing the genome, a symmetrical clone is also created, as in EVO_SYM (Figs. 5 & 6). But compared to EVO_SYM, ENF_SYM does not encode the possible symmetry in the genome. The creation of symmetrical neurons is an automatic process always occurring during the first step of the development of the neural network.

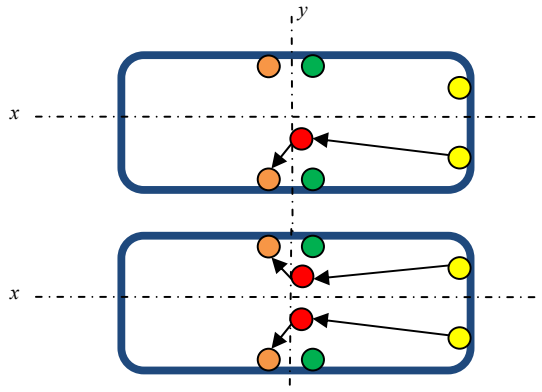

Fig. 5. At the top, the 2D substrate of an agent with the initial neural network and one intermediate neuron (in red) having two connections is shown. At the bottom, the symmetrical clone has been added.

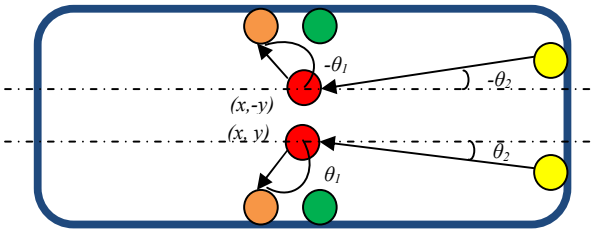

Fig. 6. Drawing showing the coordinates and the angle of the connections of the intermediate neuron (bottom) and its symmetrical clone (top).

IV. Experiments

We performed two series of tests. First, we evolved simulated agents that stay in a fixed chemical concentration. We then evolved agents to stay inside a moving concentration. In each series, we performed seven GA runs for each developmental model in order to study the importance of symmetry in neural development.

For the first series of tests, each agent had two runs of 200 seconds and started from different locations (left and right of the fixed chemical concentration). The fitness function was very simple and consisted of the sum of the inverse distances between the agent and the centre of the concentration during the last 50s of a run. The fitness of an agent was the sum of the fitness values recorded for the two runs.

For the second series of tests, we evolved agents able to stay within a moving concentration. One agent and one chemical source were placed in a toroidal world. Compared to the first series, the time of a run was longer (300s). During a run, an agent was always placed at the same place with a random angle of initial movement and the chemical concentration was placed randomly in the world. The concentration was then set moving randomly in the environment. The fitness function was also different and started to be calculated only when the agent was touching the concentration (recording time was initialized at this point). The fitness is the sum of the inverse distances, divided by the recording time. We used a resolution of 1ms (1 time step) for every simulations.

A. Genetic Algorithm

We used a classical genetic algorithm (Fig. 7) to evolve an agent that could perform chemotaxis. The initial population was composed of 100 agents. Each one of them was equipped with four motor neurons and two sensors composing the initial neural network (Fig. 2). Initially, they all had a genome composed of one module encoding one neuron, placed in the middle of the substrate, and one initial connection, having parameters randomly initialized. Therefore, the genome of these agents had one module composed of one N gene and one C gene. Then, each agent was subject to mutations (see Section IV.B). After mutations, these agents were placed in the initial population and the GA could begin. Once all the agents were evaluated, the agents were ranked by fitness and the ten fittest ones were copied to the next generation. Ninety new individuals were created and added to the next generation‘s population by selecting two parents for each, using a tournament selection of size 5. A new individual was created by cross-over of the two parents (see Section IV.B.). Out of these 90 new agents, twenty were mutated. The genetic algorithm lasted for 1000 generations.

B. Genetic Operators

The use of the following genetic operators allowed complexification of the genome by adding, modifying or removing new genes.

Mutation - In our model, mutations occur with the same probability independently of the size of the genome. Twenty agents were randomly chosen from the 90 new agents created by the tournament selection and mutated. Three kinds of mutations were used in this GA. A mutation could add or delete neurons, add or delete connections and modify the values of the parameters of the genes. Each mutation was performed within a certain range of values added to the

original ones (Tables 2 & 3), and these parameters were maintained within certain values defined earlier (Table 1). For example, if the value of the parameter x of a N gene was 49, and a mutation tried to add 5 to x (x = 54), x would be set to its maximum value 50 due to the range of values used. Here is the simple algorithm of the mutation process:

For all twenty agents:

I. Mutate each module:
   1. 5% chances to add a new connection.
   2. 5% chances to remove a randomly selected connection.
   3. Choose randomly one of the following mutations:
      - Pick randomly one connection, choose randomly one parameter and mutate it.
      - Add a random value to parameter x of a N gene.
      - Add a random value to parameter y of a N gene.

II. 5% chance to add a new module (new neuron).

III. 5% chance to remove a randomly selected module (neuron and connections).

When a new module is added to the genome, the new neuron always has one randomly initialized connection. The new neuron is placed randomly in the vicinity of the last neuron created on the substrate (last encoded in the genome). A new connection added is also always randomly initialized.

TABLE 2
RANGES OF MUTATIONS USED FOR THE PARAMETERS OF THE GENES

Parameters	Ranges of mutation
x	[-5;5]
y	[-5;5]
sym (only for EVO_SYM)	true / false
θ	[-π/4;π/4]
d	[-2;2]
w	[-5;5]
type	afferent / efferent

Cross-over - Neural selection is applied here by crossing-over modules at the same position. By doing so, each neuron should be able to specialize more quickly during evolution. Here is an example:

Two agents A_1 and A_2 are selected to create a new agent A_3. The maximum number of modules a new agent can have depends on its parents. In this case, agent A_1 has five modules and A_2 has three of them. Therefore, the new agent A_3 will have at maximum five modules (i.e. five neurons). The crossover process will make five selections of modules and at each selection, there is an equal chance of selecting the agent A_1 or A_2. Therefore, each module of the same position has 50% chances to be selected and copied. If at the fourth selection, for example, the chosen agent is A_2, which does not have any more modules at this position, nothing will be added to the genome of agent A_3 at this stage. But another module can be copied from A_1 if this one is chosen during the fifth selection, and this module, originally from position 5, will become a module of position 4 of agent A_3.

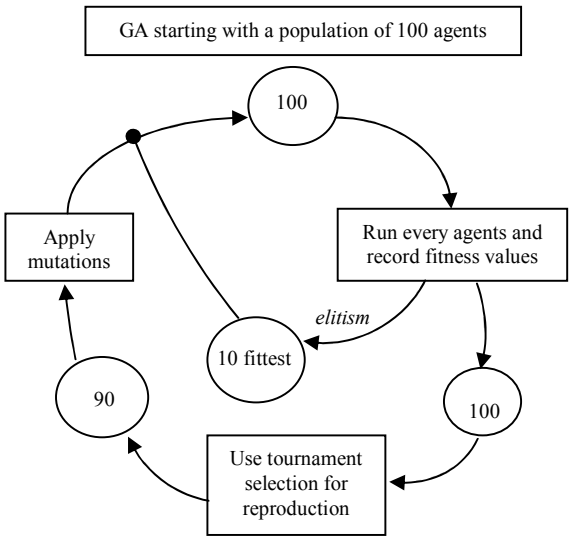

Fig. 7. Genetic algorithm with parameters. The population is composed of 100 agents. For the first experiment, each agent had two runs of 200 seconds starting from different places (left and right of the chemical concentration). In the second experiment, each agent had only one run of 300s. For both experiments, the fitness rewarded an agent that stayed inside the chemical gradient. Once all the agents were evaluated, the agents were ranked by fitness and the ten fittest ones were copied to the next generation. Ninety new individuals were created and added to the next generation‘s population by selecting two parents for each, using a tournament selection of size five. A new individual was created by cross-over of the two parents (see Section IV.B.). Out of these 90 new agents, twenty were mutated. The genetic algorithm lasted for 1000 generations. We ran the GA for 1000 generations.

V. RESULTS

In the first series of tests, we evolved agents to stay inside a chemical concentration as close as possible from its center (Fig. 3). We found that the GAs implementing the developmental models using symmetry (EVO_SYM, ENF_SYM) evolved good neural networks so the agent went in and stayed close to the centre of the concentration. We saw that in all the seven GAs, EVO_SYM evolved a neural network with symmetrical neurons. In fact, the neural controllers evolved with EVO_SYM or ENF_SYM were very similar. NO_SYM did not manage to evolve an optimal solution as the others and had an overall pretty bad performance. Therefore, the first series of tests showed us that without evolvable or enforced symmetry, the system could not evolve and find an optimal solution. In Fig. 8, we can see the neural controller of the fittest agents evolved using NO_SYM. We can clearly see that it is not bilaterally symmetric and in fact, the agent implementing it performed rather badly. Fig. 9 shows the neural network of the fittest agent evolved using EVO_SYM. This agent was performing well and used both sensors and motor neurons. It used only two symmetrical neurons where only one neuron was encoded in the genome. Neuron N1 is taking an input from

the sensor S0 and is stimulating M0 and M3 and is inhibiting M1 allowing the agent to turn quickly. N1 also has an excitatory self connection. Neuron N2 has the same symmetrical connections. We also noticed that both neurons are inhibiting each other. This neural network can be seen as an advanced Braitenberg vehicle [38]. The trajectory of this agent is shown in Fig. 3.

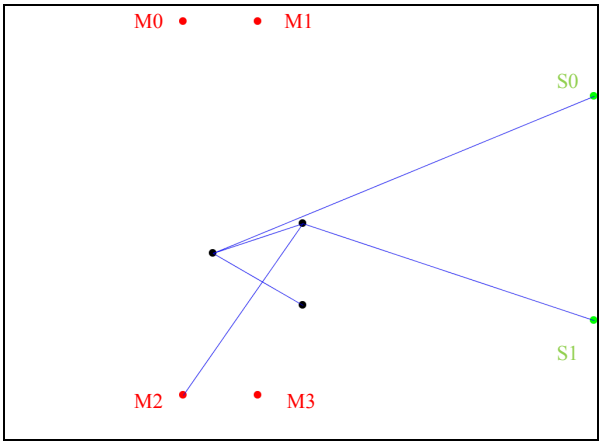

Fig. 8. Neural network of the fittest agent using NO_SYM evolved to stay in a fixed concentration. Motor neurons are depicted in red, sensory neurons in green and intermediate neurons in black.

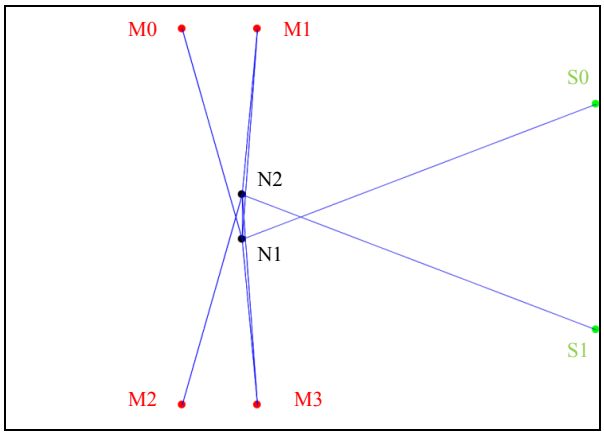

Fig. 9. Neural network of the fittest agent using EVO_SYM evolved to stay in a fixed concentration. Motor neurons are depicted in red, sensory neurons in green and intermediate neurons in black.

In the second series of test, we saw again that the developmental models using bilateral symmetry generated better neural controllers than NO_SYM (Fig. 11). Once more, EVO_SYM always evolved a neural network with symmetrical neurons in all the seven GAs. Fig. 10 and Table 3 show the neural controller of the fittest agent evolved using EVO_SYM. Neuron N1 is taking an inhibitory input from sensor S0 and an excitatory input from S1. It is stimulating M2 and is inhibiting N4 allowing the agent to turn more quickly as N4 stimulates M0. Neuron N2 is symmetrical to N1 so it has the same symmetrical connections. We also noticed that both neurons are inhibiting each other. Neuron N3 takes input from the sensor S1 and stimulates the motor neurons M1 and M2 so the agent can turn quickly. Neuron N4 is symmetrical to N3 so it has the same symmetrical connections. We notice that two other symmetrical neurons (N5 and N6) and a non symmetrical neuron (N7) exist but they do not modify the overall neural activity of the controller. This shows that symmetrical neurons (N1 and N2) can also have asymmetrical connections. N5, 6 and 7 can be seen as evolutionary artefacts that could become useful in time or disappear. This neural network has more complexity that the one shown in Fig. 9. The main differences between the two are the two layers of neurons and inhibitory connections coming from the sensors. We also noted that neurons N3 and N4 created more than one connection to the motor neurons. We suppose that it is due to the limit values the weights can have [-15; 15] (see Table 1). Therefore, we can see that the system can easily adapt to circumvent certain constraints.

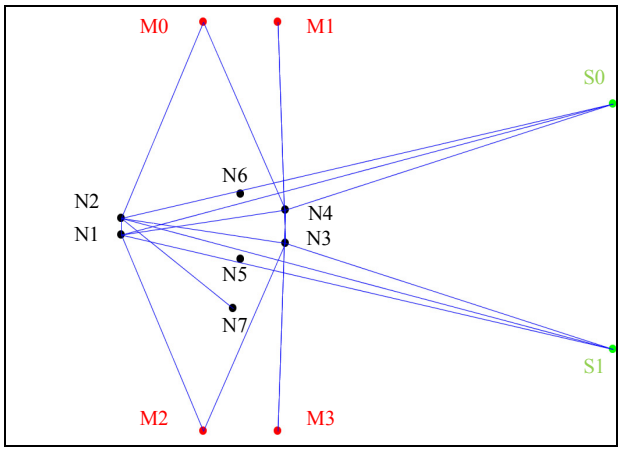

Fig. 10. Neural network of the fittest agent using EVO_SYM evolved to stay in a moving concentration. Motor neurons are depicted in red, sensory neurons in green and intermediate neurons in black.

TABLE 3
WEIGHT MATRIX OF THE NN FROM FIG. 9 SHOWING THE CONNECTIONS LINKING CELLS (TOP ROW) TO OTHER CELLS (LEFT COLUMN)

Cells	S0	S1	N1	N2	N3	N4	N7
M0				15		15	
M1					6, 15		
M2			15		15		
M3						6, 15	
N1	-6	9		4, -15			
N2	9	-6	-15				4
N3		6		-10			
N4	6		-10				
N7							-8

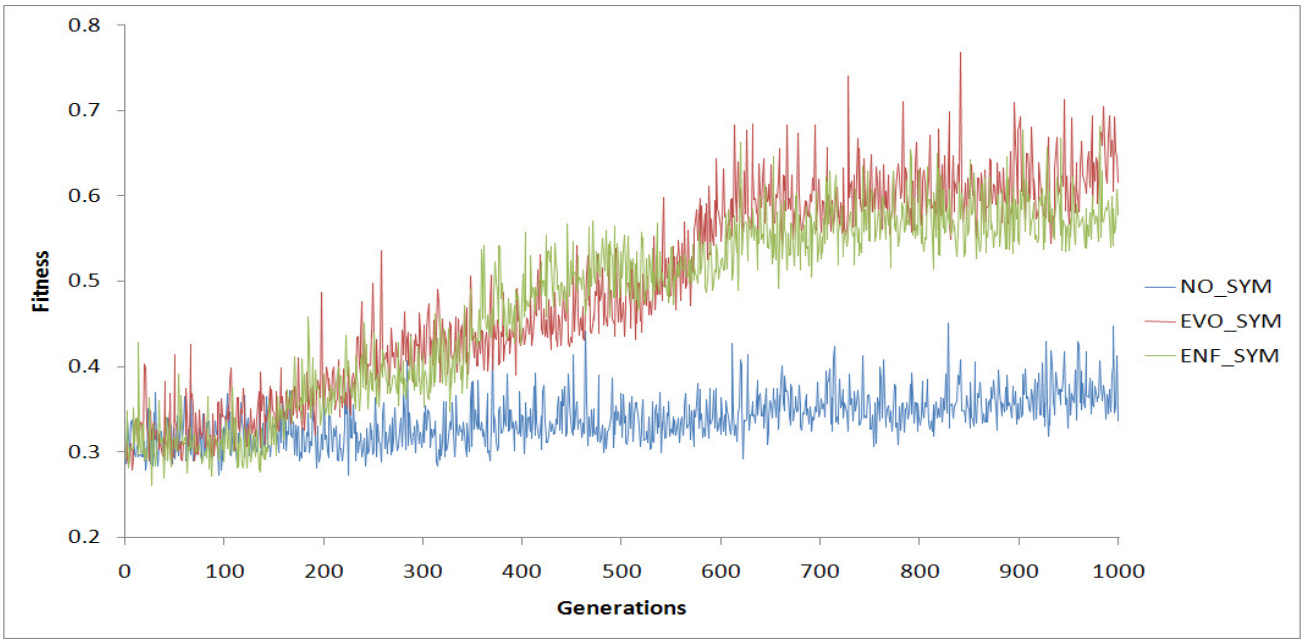

Fig. 11. Fitness mean values over seven GA runs of fittest agents per generation. The agents were evolved to perform chemotaxis with a moving target. This graph shows that the use of bilateral symmetry (EVO_SYM and ENF_SYM) created neural controllers performing considerably better than without symmetry (NO_SYM).

VI. Discussion

In this paper, we have shown that bilateral symmetric neural networks can be evolved using a genetic algorithm and our developmental models, and have better performances than non-symmetrical ones. Perhaps this is not surprising. Firstly, the agent has bilaterally placed sensors and actuators. Secondly, the task of chemotaxis also has implicit symmetry: a chemical to the left triggers a turn to the left and symmetrically, a chemical to the right triggers a turn to the right.

Complexification, targeting and neural selection are important concepts in our model. We use a 2D neural substrate where spiking neurons are placed and can grow connections to target locations. Therefore, the geometric configurations of the neural network significantly matter. Since we use spiking neurons with conduction delays, distances separating connected neurons encode time delays between the points in time spikes are sent by a neuron, and the time they are received by another neuron. A neural network generated by our developmental models can encode information not only using firing rate encoding but also using temporal coincidence or delay encoding [34, 35, 39]. Evolution can therefore generate neural networks able to encode external information as spatio-temporal patterns. More detailed analysis of the activity of the different neural networks that evolved will be done in the future to see which neural encodings were really used.

We have noticed from our results that sometimes more than one connection linking two cells was created. This is due to the limits of the weights used, showing that the system can easily adapt to certain limiting constraints. We have seen that connections between symmetrical parts of the neural controller could be connected and inhibitory connections of symmetrical neurons were often evolved. Also, neural controllers grown with NO_SYM could have symmetrical neurons, but did so with an extremely low probability.

We have to emphasize the fact that the initial neural network, placed on the substrate, is bilaterally symmetrical. Most physical robots are also bilaterally symmetric, and therefore, we assume that mapping sensors and motors to sensory and motor neurons on the neural substrate could be done in a direct manner when implementing our model on a simulated and real robot. In this case, it biased evolution to find an appropriate solution that uses this embedded symmetry. It would be very interesting to see if bilateral symmetry would still arise and be beneficial when evolving the morphology of the agent as well as the neural substrate. Cells could migrate on the substrate and differentiate to become sensors, neurons and motor neurons.

Many modifications of this model can be done. For example, adding the possibility to encode the threshold of a neuron or different axes of symmetry in the genome. Other developmental models could have been created where only one gene could have created symmetrical neurons for the entire neural network. However, we decided to use EVO_SYM to permit the creation of both symmetrical and asymmetrical parts, and therefore to increase complexity.

VII. Conclusion

In this paper, we have presented three novel developmental models allowing information to be encoded in space and time using spiking neurons placed on a 2D substrate. In two of these models, we introduce neural development that could use bilateral symmetry. We showed

that these models created neural controllers that permit agents to perform chemotaxis, and do so better than controllers with no symmetry. We also have showed that with EVO_SYM, neural bilateral symmetry was often evolved and was found to be beneficial for the agents. This work is the first, as far as we know, to present developmental models where spiking neurons were generated in a 2D space and where bilateral symmetry could be evolved and was proved to be beneficial in this context.

In future work, we will use incremental evolution to generate agents that can perform more than one task. Our long term interest is to study the emergence of chemical communication in a population of artificial agents.

REFERENCES

[1] D. A. Thompson, *On Growth and Form*: Cambridge University Press (1992), 1917.

[2] H. Meinhardt, *Models of Biological Pattern Formation*: Academic Press, 1982.

[3] H. Weyl, *Symmetry*: Princeton Press, 1952.

[4] M. Enquist and R. A. Johnstone, "Generalization and the evolution of symmetry preferences," *Proceedings of the Royal Society, Biological Science,* vol. 264, pp. 1345-1348, 1997.

[5] W. Arthur, *The Origin of Animal Body Plans, A Study in Evolutionary Developmental Biology*: Cambridge University Press, 1997.

[6] S. F. Gilbert, *Developmental Biology, Eighth Edition*, 8 ed.: Sinauer Associates, 2006.

[7] A. R. Palmer, "Symmetry Breaking and the Evolution of Development," *Science,* vol. 306, pp. 828-833, 2004.

[8] L. Wolpert, "Development of the asymetric human," *European Review,* vol. 13, pp. 97-103, 2005.

[9] Y. Dongyong and S. i. Yuta, "Evolving Mobile Robot Controllers Using Evolutionary Algorithm," in *SICE 2002, Proceedings of the 41st SICE Annual Conference*. vol. 4, 2002, pp. 2184-2189.

[10] D. Floreano, P. Husbands, and S. Nolfi, "Evolutionary Robotics," in *Springer Handbook of Robotics, Chapter 61*, B. Siciliano and O. Khatib, Eds.: Springer, 2008.

[11] R. Pfeifer, F. Iida, and J. Bongard, "New Robotics: Design Principles for Intelligent Systems," *Artificial Life, Special Issue on New Robotics, Evolution and Embodied Cognition,* vol. 11, pp. 99-120, 2005.

[12] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence,* vol. 1, pp. 47-62, 2008.

[13] E. Ruppin, "Evolutionary Autonomous Agents: A Neuroscience Perspective," *Nature Reviews Neuroscience,* vol. 3, pp. 132-141, 2002.

[14] X. Yao, "Evolving Artificial Neural Networks," *Proceedings of the IEEE,* vol. 87, pp. 1423-1447, 1999.

[15] S. Nolfi, O. Miglino, and D. Parisi, "Phenotypic Plasticity in Evolving Neural Networks," Technical Report PCIA-94-05, Institute of Psychology, C.N.R., Rome 1994.

[16] A. Channon and R. I. Damper, "The Evolutionary Emergence of Socially Intelligent Agents," in *Socially Situated Intelligence: a workshop held a SAB'98*, 1998.

[17] A. Channon and R. I. Damper, "Evolving Novel Behaviours via Natural Selection," in *Proceedings of the 6th Conference on the simulation and synthesis of living systems (ALIFE VI)*, Los Angeles, CA, USA, 1998, pp. 384-388.

[18] T. Miconi and A. Channon, "A virtual creatures model for studies in artificial evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, UK, 2005.

[19] T. Miconi and A. Channon, "An Improved Sytem for Artificial Creatures Evolution," in *Proceedings of the 10th Conference on the simulation and synthesis of living systems (ALIFE X)*, Bloomington, Indiana, USA, 2006.

[20] G. S. Hornby and J. B. Pollack, "Body-Brain Co-evolution Using L-systems as a Generative Encoding," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 2001.

[21] K. Sims, "Evolving 3d morphology and behavior by competition," in *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems (ALIFE IV)*, 1994, pp. 28-39.

[22] M. Komosinski and A. Rotaru-Varga, "Comparison of Different Genotype Encoding for Simulated 3D Agents," *Artificial Life,* vol. 7, pp. 395-418, 2001.

[23] F. Gruau, "Genetic synthesis on modular neural networks," in *Proceedings of the 5th International Conference on Genetic Algorithms*, San Francisco, CA, USA, 1993, pp. 318-325.

[24] J. Gauci and K. O. Stanley, "Generating Large-Scale Neural Networks Through Discovering Geometric Regularities," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*: New York, NY: ACM, 2007.

[25] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems,* vol. 8, pp. 131-162, 2007.

[26] D. Filliat, J. Kodjabachian, and J. A. Meyer, "Incremental evolution of neural controllers for navigation in a 6 legged robot," *Proceedings of the Fourth International Symposium on Artificial Life and Robotics,* vol. In Sugisaka and Tanaka, editors, Oita Univ. Press., 1999.

[27] A. Ijspeert and J. Kodjabachian, "Evolution and development of a central pattern generator for the swimming of a lamprey," *Research Paper No 926,* vol. Dept. of Artificial Intelligence, University of Edinburgh, 1998.

[28] J. Kodjabachian and J. A. Meyer, "Evolution and Development of Neural Controllers for Locomotion, Gradient-Following, and Obstacle-Avoidance in Artificial Insects," *IEEE transactions on neural network,* vol. 9, 1998.

[29] J. Kodjabachian and J. A. Meyer, "Evolution of a Robust Obstacle-Avoidance Behavior in Khepera: A Comparison of Incremental and Direct Strategies," *IEEE transactions on neural network,* vol. 9, 1997.

[30] J. Kodjabachian and J. A. Meyer, "Evolution and development of modular control architectures for 1-d locomotion in six-legged animats," 1997.

[31] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation,* vol. 10, pp. 99-127, 2002.

[32] K. O. Stanley and R. Miikkulainen, "A Taxonomy for Artificial Embryogeny," *Artificial Life journal,* vol. 9, pp. 93-130, 2003.

[33] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science*, 4th ed.: McGraw-Hill, 2000.

[34] N. Oros, V. Steuber, N. Davey, L. Cañamero, and R. G. Adams, "Adaptive Olfactory Encoding in Agents Controlled by Spiking Neural Networks," in *Proceedings of The 10th international conference on Simulation of Adaptive Behaviour, From Animals to Animats (SAB)*, Osaka, Japan, 2008, pp. 148-158.

[35] N. Oros, V. Steuber, N. Davey, L. Cañamero, and R. G. Adams, "Optimal noise in spiking neural networks for the detection of chemicals by simulated agents," in *Proceedings on the Eleventh International Conference on Artificial Life*, Winchester, UK, 2008.

[36] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of Brownian Motion," *Physical Review,* vol. 36, pp. 823-41, 1930.

[37] N. Oros, V. Steuber, N. Davey, L. Cañamero, and R. G. Adams, "Optimal receptor response functions for the detection of pheromones by agents driven by spiking neural networks," in *19th European Meetings on Cybernetics and Systems Research*, Vienna, 2008.

[38] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*: MIT Press, Cambridge, MA., 1984.

[39] D. Floreano and C. Mattiussi, "Evolution of Spiking Neural Controllers for Autonomous Vision-Based Robots," *Proceedings of the international Symposium on Evolutionary Robotics From intelligent Robotics To Artificial Life,* vol. 2217, T. Gomi, Ed. Lecture Notes In Computer Science Springer-Verlag, London, 38-61., 2001.