Proceedings of the 2007 IEEE Symposium on
Foundations of Computational Intelligence (FOCI 2007)

SOUND LOCALIZATION THROUGH EVOLUTIONARY LEARNING APPLIED TO SPIKING NEURAL NETWORKS

Thomas M. Poulsen, Roger K. Moore

Speech and Hearing Group, Department of Computer Science, University of Sheffield

ABSTRACT

A biologically based learning framework is established to study neural modeling with respect to sound source localization. This involves a 2-dimensional environment wherein agents must locate sound sources that are periodically resituated whilst emitting pulses at regular intervals. Agents employ a spiking neural model that controls movement on the basis of binaural inputs, and evolutionary learning (EL) is applied to evolve neural connectivity and weights. It is demonstrated that agents are successfully able to locate sound sources and that the simulative framework can be extended to address questions pertaining to the evolution of spiking neural networks.

1. INTRODUCTION

Organisms that display sophisticated behaviour provide a framework for investigating models of intelligence and the neural functionality that underpins such behavior. However, deciphering brain functioning is conceptually similar to reverse-engineering the structure of a complex piece of electronics; it becomes increasingly difficult with more advanced designs. It is therefore helpful to study simpler models which, despite dissimilarities, still utilize many of the same fundamental building blocks found in more complex systems. By looking at sequential items along the same development process, the steps and decisions through various stages of advancement in complexity may explain aspects of more sophisticated designs. Likewise, organisms with simpler brains than humans (albeit still very complicated) are part of an advancement process that can provide insights throughout the various stages of evolutionary development.

One approach to understanding brain functionality is therefore to investigate the composition and signaling in simple neural structures that evolve into more complicated designs. Simulations that employ an evolutionary process provide a framework for such an investigation and a number of works have successfully approached the evolution of neural networks [1-4]. Such simulations have provided insights into agent behavior with respect to neural development. They however make use of rate based neural models (models that do not take temporal effects of neurons into account) although it has been shown that sensory stimuli are temporally encoded by neurons [5-9]. In contrast, the simulations presented here employ a biologically-based spiking neural model which takes temporal behavior into account.

The work reported in this paper constitutes part of a larger investigation into a simulative framework for the evolution of biologically based neural models by using evolutionary learning (EL). The simulation described here is intended to provide an important proof of concept to demonstrate that EL can be used to successfully evolve a spiking neural network for agents that receive stimuli in a simulative environment. The successful evolution of a spiking neural network would support the use of simulative work for analysing neural connectivity and signalling, in particular the temporal encoding of stimuli. This paper focuses on the evolution of neural networks for the task of sound localization.

Sound localization is an aspect of animal behaviour that can be critical in situations like those between predator and prey, parents and offspring, and potential mates. It is therefore an important behavioural attribute in animals and it also provides a simple yet effective simulative framework for testing agents in different scenarios. The simulation described below is inspired by that of Werner and Dyer [1], where blind male agents must locate immobile female agents based on incoming signals. It however differs in a number of important respects. Particularly, in W&D's work, male and female agents communicate, agents meet to reproduce, and the simulation employs rate based neural networks. In the simulation reported here, a genetic algorithm is used to select and produce offspring, agents locating sound sources cannot signal, and spiking neural networks have been employed. This work thus provides a greater focus on the development of the spiking neural network in a simulative environment.

2. SIMULATIVE ENVIRONMENT

The overall scenario is that a community of agents must locate sound sources that emit a simulated acoustic pulse once every 10 msecs. In order to evaluate agent performance at finding sources in different directions, each source is randomly repositioned every two seconds over a period of 10 seconds. When an agent successfully reaches a

source it is randomly resituated on the map for a new attempt.

Similar to Werner and Dyer's simulation, agents populate an environment that consists of a two-dimensional world that is 200 x 200 units in size. The world is flat and constrained by imaginary walls. This constrained arrangement (as opposed to an unconstrained round world) means that agents do not develop the trivial behavior of constantly moving in a random single direction, which can falsely make it seem like an agent sometimes moves closer to a source.

2.1. Agent Neural Structure

Each agent possesses an artificial neural network that receives input and controls agent actions. The neural network is defined by a genome consisting of input neurons, hidden neurons, and output neurons. These neuron types are allocated according to genome makeup and neurons are fully connected within a prior set of constraints: input neurons act as sensors and do not receive any neural inputs, whilst output neurons control agent movement and receive connections from input and hidden neurons. Hidden neurons function as an adjoining layer between input and output neurons, and they connect to one another as well as to output neurons. The connections between neurons can be either excitatory or inhibitory, and are specified through genome data.

The genome consists of real numbers that are divided into sections pertaining to one of two neural aspects: a neuron's connections or its type (input, hidden or output). As such, each neuron has a 'connection' and a 'type' section in the genome, and the genome can be sequentially divided according to neural indices.

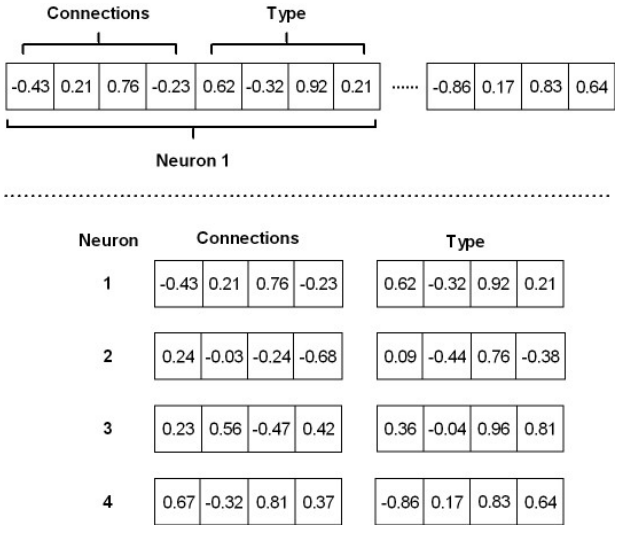

Figure 1 – Genome representing neural structure consisting of four neurons.

An example of this structure is depicted in figure 1, which shows a network consisting of four neurons. Neuron 1 receives connections from neurons 1, 2, 3 and 4 with corresponding weights of -0.43, 0.21, 0.76, -0.23. A neuron is however not allowed to connect to itself, so the connection from neuron 1 to itself is ignored.

A neuron's type is decided by the following formula:

$$Neurontype = S_n \bmod T$$

where T is the number of neural types and in the simulation $T = 3$. S_n is given by summating genome values for a given neuron n:

$$S_n = \sum_i Round(|10 \cdot g_i|)$$

The genome value at position i is depicted by g_i, which is multiplied by ten to attain sufficiently high values. In the case of figure 1, the type of neuron 1 is for example calculated with g_i values of 0.62, -0.32, 0.92, 0.21. Absolute values are used, which are then rounded off. Neuron types are then classified according to the following:

$$Neurontype = 0 \Rightarrow Input\ neuron$$
$$Neurontype = 1 \Rightarrow Hidden\ neuron$$
$$Neurontype = 2 \Rightarrow Output\ neuron$$

For the sake of simplicity, and to provide a suitable amount of flexibility for genome change during crossovers and mutations, the number of 'type' values in the genome is set to the number of 'connection' values for each neuron (as in figure 1). The genome length is therefore $2N^2$, where N is the total number of neurons in each neural network.

The genome was designed with a simple genetic algorithm in mind and it should be noted that more optimal designs could be derived to reduce the genetic algorithm's search space. Likewise, a simple crossover function was employed but utilizing a more sophisticated crossover method designed for real values would also improve GA efficiency. Although the genome design and crossover function sufficed for the needs of the simulation, a more efficient algorithm could as such be derived.

3. SPIKE RESPONSE MODEL

The spike response model [10] (SRM) has been selected to model neural firing dynamics. SRM has shown that it can successfully capture many of the dynamic behaviors of biological neurons [10,11], and its kernel flexibility makes it adaptable to the agent scenario. Floreano and Mattiussi [12] have also demonstrated the applicability of evolutionary learning to an SRM network, although their

work involves development of weights in a statically connected neural network for vision-based navigation of a robot. Nevertheless, F&M's model selections have provided a useful basis to follow.

The functionality of the SRM investigated here is explained below, but for an understanding of the model's relation to the underlying biological substrate, the reader is referred to Kistler et al. [10].

The SRM can be thought of as a generic 'integrate and fire' model and it consists of different kernels that have been derived on the basis of neural firing characteristics. In the case of the simulation described here, membrane potentials are calculated depending on whether a neuron receives sensory input (input neurons) or not (hidden and output neurons). The membrane potential of a neuron at time t is calculated using the following spike response models:

Input neurons: $u_i(t) = \eta(t - \hat{t}_i) + \int_0^\infty \kappa(t - \hat{t}_i, s) I^{ext}(t - s) ds$

Other neurons: $u_i(t) = \eta(t - \hat{t}_i) + \sum_j w_{ij} \sum_f \varepsilon_{ij}(t - t_j^{(f)})$

where η, κ, and ε are the kernels mentioned, and j is used to index the presynaptic neurons that connect to neuron i. The same η and ε kernels employed by Floreano and Mattiussi were chosen, and κ was selected from Gerstner and Kistler [11].

$$\eta(s) = \eta_0(s) = -(u_{thresh} - u_{rest}) e^{-\frac{s}{\tau_m}}$$

where η describes the effects on the membrane potential after a neuron has generated a spike, $\hat{t}$ refers to the last time it fired, u_{thresh} is the membrane threshold, and u_{rest} is the membrane resting potential. A neuron's membrane potential is reset to its resting potential after it has fired. This effect negates such that the neuron 'recovers' over a time period that is dependent on the membrane time constant τ_m.

The effect of incoming spikes declines over time such that more recent spikes incur greater influence on the membrane potential. This effect is dependent on the properties of the neural membrane and the synaptic connection (time constants τ_m and τ_s respectively) and is expressed by:

$$\varepsilon(s) = e^{-\frac{s - \Delta_{abs}}{\tau_m}} (1 - e^{-\frac{s - \Delta_{abs}}{\tau_s}}) \Theta(s - \Delta_{abs})$$

where ε as such expresses the response to presynaptic input and is dependent on the last firing times t^f_j of the presynaptic neurons. Spikes coming from presynaptic neurons do not arrive immediately, but after a time delay of Δ_{abs}. This is enforced by the Heaviside step function Θ: $\Theta(x) = 1$, $x > 0$ else $\Theta(x) = 0$.

The external current I^{ext} that input neurons receive, contributes with an increase in membrane potential, which has a decay rate that is dependent on when the neuron last fired. Neural membrane potential quickly drops after a spike, and the input I must not be able to disrupt this mechanism. Otherwise, a neuron could potentially fire without any refraction time, which in biological neurons would be impossible. These dynamics are dependent on the decay factor e^{-s/τ_m} and the recovery response $(1 - e^{-(t-\hat{t})/\tau_{rec}})$.

$$\kappa(x, s) = \frac{R}{\tau_m} (1 - e^{-\frac{x}{\tau_{rec}}}) e^{-\frac{s}{\tau_m}} \Theta(x - s) \Theta(s)$$

where κ describes a linear response in membrane potential to an external input current with an input resistance of R ($s = t - t^f_i$)and is included to simplify the integral factor).

Based on Gerstner and Kistler [11] and initial experimental results, simulation parameters were set to:

$$R = 1$$
$$\tau_m = 0.02$$
$$\tau_s = 0.0015$$
$$\tau_{rec} = 0.0125$$
$$\Delta_{abs} = 0.002$$
$$u_{thresh} = 0.5$$
$$u_{rest} = 0$$

Membrane potentials were calculated using an iterative time step of 1 msec. As such, one iteration in the simulation corresponds to this value.

4. AGENT PROPERTIES

Agents in this simulation were not provided with any visual sensors and only pulse signals are received as input. Movement is based on two wheels that each have one axis of freedom. Agents thus turn by adjusting relative speeds of the wheels. When an agent moves, it is always at a constant speed of one unit per 20 msec, regardless of how fast the wheels turn.

4.1 Agent Motor Control

Agent neural structures consist of N neurons, which is an experimental variable in the simulation. An agent's wheels are controlled by the output neurons where one half of the neurons is connected to the left wheel and the second half is connected to the right. If there are an odd

number of neurons, then one output neuron is disregarded. Following the approach of Floreano and Mattiusi [12], wheels are turned on the basis of the spike rate to each wheel in a 20 msec timeframe. Agents do not move if each wheel receives zero spikes in the timeframe, and if only one wheel receives spikes, then the agent turns but does not move forward. If both wheels receive spikes, the agent turns and moves at a constant speed of one unit. When turning, each wheel's turning speed is proportional to the amount of spikes it receives denoted by n_l and n_r for the left and right wheel respectively. A maximum turning speed is invoked for each wheel by the constant M : if n_l or n_r are greater than M they are set to M. In practice, this works as a normalizing factor when calculating the new direction d (where $0 \le d \le 2\pi$):

$$d = (d + 2\pi \frac{n_r - n_l}{M}) \bmod 2\pi$$

If d is negative, it is normalized to bring it within 0 and 2π .

4.2 Agent Sensor Input

Agents possess two sensors with which they can receive pulse signals. These can be thought of as ears that are placed on opposite sides of the agent, at an equal distance from its centre (see Figure 2). The absolute positions of the ears in the environment are calculated every time an agent moves. Given that an agent has a direction d , then the ears are positioned with relative angles to d such that:

$$\text{Angle of left ear} = d - \frac{\pi}{2} \quad \text{Angle of right ear} = d + \frac{\pi}{2}$$

The x and y coordinates of the ears (relative to the centre of the agent) can then calculated using basic trigonometry:

$$x_{left} = a\cos(\theta_{left}),\ y_{left} = a\sin(\theta_{left})$$
$$x_{right} = a\cos(\theta_{right}),\ y_{right} = a\sin(\theta_{right})$$

where a can be used as a scaling factor to adjust the distance between the ears. In the simulations one unit in the environment is considered to be 1 meter in length and signals move at the speed of sound (≈340.0 m/s). Thus, if no scaling factor is utilized between the ears, they will have a distance of 2 meters apart and a time delay between them of up to about 6 msec. Alternatively if *a=0.1* then the ears are only 0.2 meters apart providing a more realistic scenario.

Interaural time differences (ITDs) are calculated according to the position of the ears relative to the sound source (see Figure 2). For example, if an agent's left ear is closest to the source, it will receive a pulse emitted at time t depending on its position. The signal will arrive at the right ear at t + delta, where delta is the ITD that would exist between the ears.

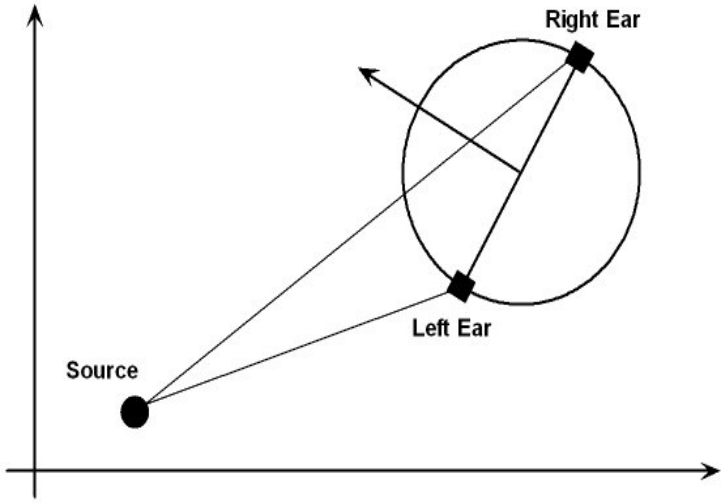

Figure 2 – Illustrative example of how ears are positioned with respect to an agent's direction (denoted by the arrow) and a sound source.

Interaural intensity differences (IIDs), also termed interaural level differences (ILDs) in some literature, have been highly simplified and are based on a simple scaling factor c (where $c \ge 1$). Given a signal strength of I_{base} then the closest ear to the source receives a signal strength of:

$$I_{closest} = \frac{I_{base}}{\sqrt{x_{closest}}}$$

where $x_{closest}$ is the distance from the ear to the source. The ear farthest away receives:

$$I_{farthest} = \frac{I_{base}}{c\sqrt{x_{farthest}}}$$

If the ears are almost the same distance to the source then the signal strength for both ears is calculated such that signals coming from in front are louder than those coming from behind:

$$I_{front} = \frac{I_{base}}{\sqrt{x_{front}}}, I_{behind} = \frac{I_{base}}{c\sqrt{x_{behind}}}$$

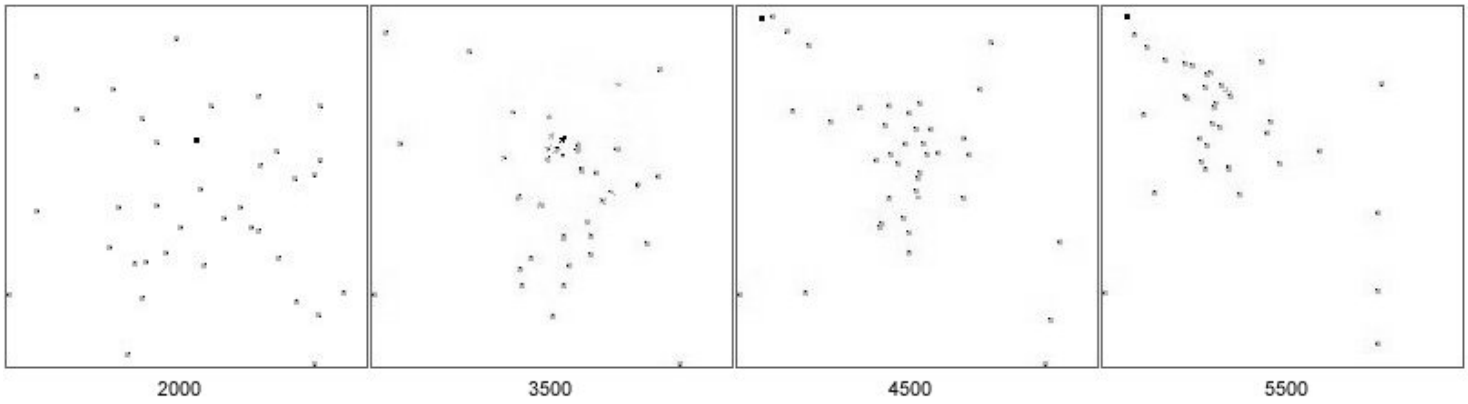

Figure 3 – Agent positions at four different iterations in a simulation where agents must locate a single sound source (black square). The source has been repositioned at 2000 and 4000 iterations, and the majority of agents can be observed to successfully move directly towards the source at its new positions. The small black points on agents indicate their approximate direction.

These values are then directly inserted as I^{ext} when calculating membrane potentials for input neurons.

4.3 Evolutionary Learning

Agents are evaluated every 20 msecs (the timeframe for agents to move their wheels) exclusively on the basis of how much closer they have moved to the nearest source. Reaching the source does not offer any additional reward. An agent's fitness score is then calculated by summing the scores over all evaluated iterations. Each iterative score is calculated depending on whether an agent moved closer to the source or not:

$$\text{Moved closer: } v_n = (d_2 - d_1)^2$$

$$\text{Idle or moved farther away: } v_n = -(d_2 - d_1)^4$$

where d_1 is an agent's distance to source before moving and d_2 is the distance after moving, for the nth evaluated iteration. As agents always move at a constant speed of one unit, then $|d_1 - d_2| \leq 1$. Given a total of Z iterations the fitness score f of an agent can then be calculated as:

$$f = \frac{\sum_n v_n}{Z}$$

It is possible for agents to receive negative scores since they are penalized for moving further away. In this case the fitness score is set to zero. Agents were sometimes observed to move in small circles or back and forth (and thus receiving reasonable fitness scores), which prompted the introduction of a lesser penalty score for moving in the wrong direction. This type of scoring seemed to reduce the frequency of such behavior from early generations onwards.

5. EXPERIMENT

5.1. Experimental Set-up

In the simulations, single crossovers were used with a crossover rate of 0.15 and a mutation rate of 0.07, and the population size was set to 40.

Simulations were made with neural structures consisting of 15 neurons. This number was decided on through early experimental runs where results indicated this to be an approximate optimum for the simulations. Typically agents that employed models with less than 8 neurons would not perform well and increasing the number above 15 neurons did not appear to improve results. It should be noted though, that this figure was not explored with respect to a large variation of parameters.

To affirm the effects of IIDs and ITDs, simulation runs where made with only IIDs (by setting $a = 0$, ITDs are always zero) and with only ITDs (with $c = 1$, IIDs become zero). Different ranges of IIDs and ITDs were also tested, typically with ears 0.2 to 2.0 meters apart, and an IID scaling factor between 1.25 and 2.0 was applied (and $I_{base}=1$). Simulations using 1, 2, and 4 different sound sources were made to observe agent performance with respect to multiple sources. All simulation runs were for 100 generations.

Videos of simulations can be found at: http://www.dcs.shef.ac.uk/~thomas/videos.html

5.2. Single Sounds Source Results

5.2.1. Simulations with both ITDs and IIDs

In the first simulation experiment, agents had to find a single sound source in the environment. Simulations were made using different scaling factors for IIDs and ITDs, and

the following values were employed:

$$a = 0.1, 0.5, 1.0$$
$$c = 1.25, 1.5, 2.0$$

Agents trying to locate a single sound are illustrated in figure 3. The sound source is repositioned at 2000 iterations, and as can be observed, most agents have gathered around the source at 3500 iterations. The source is repositioned again at 4000 iterations, and at 5500 iterations, the progress of many agents can be observed. It is important to note that some agents are not close to the source either because it was positioned far away from them or because they had already reached it (and thus had been repositioned on the map).

Agents were overall very successful in reaching a single source, with the best agents reaching fitness values up to about 0.9. Figure 4 illustrates such a simulation where a = 1.0 and c = 2.0. Performance was affected slightly by using lower values of a and c such that fitness values usually decreased overall by about 0.05. These values reflect very good performances, especially considering the relatively stringent evaluations that have been applied. The evolutionary progress of agents can be observed in figure 4, where agent fitness values reached a plateau after approximately 30 generations. This was typical for most simulations.

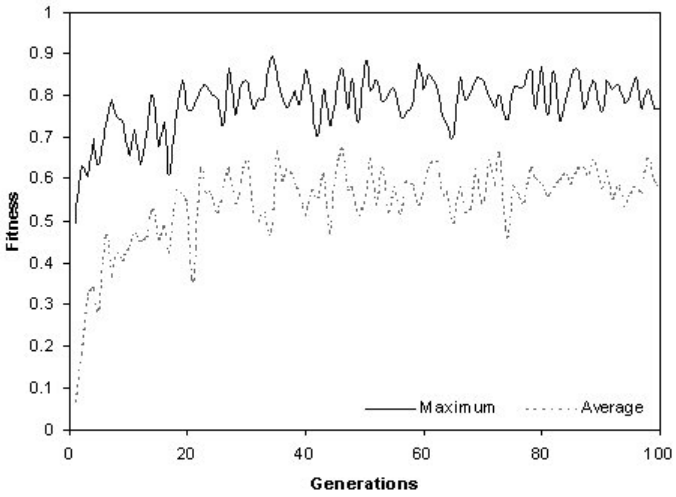

Figure 4 – Agent fitness development in an environment with one sound source (a = 1.0, c = 2.0). Fitness values increase until approximately 30 generations after which values level out.

5.2.2. Simulations with either ITDs or IIDs

To verify that ITDs and IIDs were indeed being utilized by agents, simulations were made where agents could only detect one of the two. The following scaling factors were used:

No ITDs: a = 0.0; c = 1.25, 1.5, 2.0
No IIDs: c = 1.0; a = 0.1, 0.5, 1.0

Figure 5 depicts the results from a simulation where agents are only able to detect IIDs. As can be seen, this affected results considerably. Similar scores to those in figure 5 were produced when only ITDs existed. These results demonstrate that agents exploited IIDs and ITDs, and that they were important to their success.

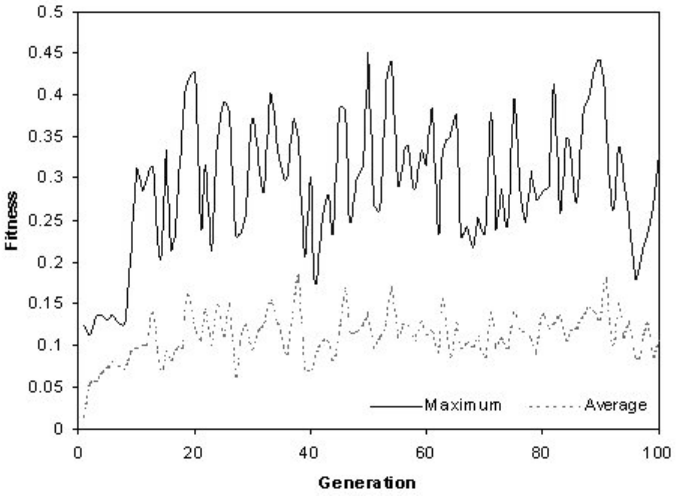

Figure 5 – Results from a simulation with a single sound source where agents were only able to detect IIDs. It is interesting to note the decline in results compared with those in figure 4, where agents could detect both ITDs and IIDs.

5.3. Multiple Sounds Source Results

Locating multiple sound sources is a considerably more difficult task than a single source, and it is therefore an interesting experiment to observe if agents can locate the correct sound source (where the *correct* source is the closest source to an agent). Every source emits signals with equal strength I_{base}, which reach agents at a given time based on their position (as described in section 4.2). There is a small likelihood that signals will arrive at the same time given iterations of 1 msec in the simulation, but in such an event, they are simply processed as separate neural inputs.

Looking at figure 6 it can be observed that agents were able to perform very well with two sound sources in the environment. However, observing agent movements revealed that many of them did not move to the closest source, but instead seemed to choose arbitrarily. This would account for the slightly lower fitness scores (compared with a single source), since agents would be penalized until they got close enough to the source that they were moving towards. Simulations with four sound sources provided lower results such that fitness values were overall approximately 0.1 less than fitness scores with two sound sources. Not surprisingly, introducing more sound sources

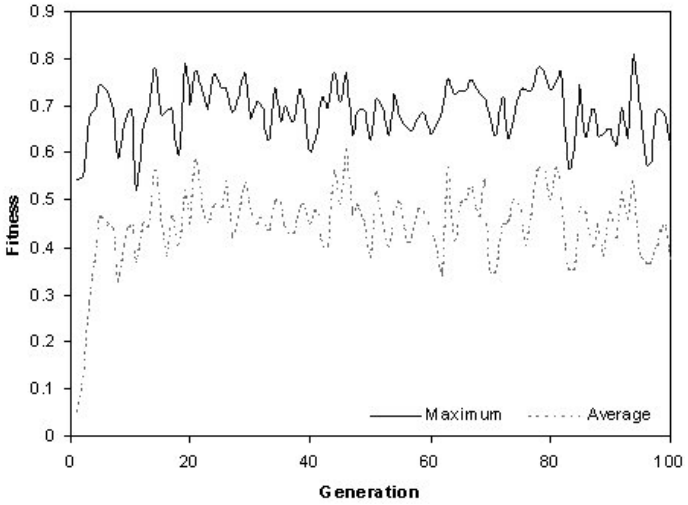

Figure 6 – Agent fitness scores in a simulative environment with two sound sources. Fitness decreased overall by approximately 0.05-0.10 compared with agents attempting to locate a single sound source.

into the environment confuses agents, but it is interesting to note that agent performances are still respectable.

6. CONCLUDING REMARKS

Most agents were able to successfully locate single sound sources and reached high fitness values of approximately 0.9. Introducing multiple sound sources prompted lower fitness scores, and agents did not always move to the correct source. Yet agents were still generally capable of locating a sound source despite this added confusion. The experimental results overall demonstrated successfully that the simulation framework was able to evolve neural models for sound localization.

Implementing features like acoustic representation and more accurate calculations of ITDs and IIDs would provide a further realistic simulative framework that could be used for investigating questions that relate to sound localization. Simulations could then be used to study the neural encoding involved in sound localization, building full auditory systems, or investigating more specific aspects like how neurons factor out reverberation from direct sounds.

The investigation reported here was conducted in the context of a wider research programme into the evolution of neural networks and neural encoding of signals for communication. Follow-up research is extending the simulation to more realistic scenarios of sound localization in order to provide the basis for a neuroethological analysis. Such an analysis (e.g. Floareano and Mattiusi [12]) encompasses aspects such as measuring the significance of synaptic channels and evaluating the roles of individual neurons. We anticipate that this line of research will provide insights into neural encoding and structuring in agent-based communication.

7. REFERENCES

[1] G. M. Werner and M. G Dyer, "Evolution of communication in artificial organisms", In *Proceedings of the Workshop on Artificial Life*, pages 659-687, Addison-Wesley, 1991.

[2] R. D Beer and J. Gallagher, Evolving dynamic neural networks for adaptive behavior. *Adaptive behavior*, 1(1):91-122, 1992.

[3] R. D Beer, "Toward the evolution of dynamical neural networks for minimally cognitive behavior", In *Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, 421-429T, MIT Press, Cambridge MA, 1996.

[4] E. A. Di Paolo, "Behavioral coordination, structural congruence and entrainment in a simulation of acoustically coupled agents", *Adaptive Behavior*, 8(1): 27-28, 2000.

[5] F. Gabbiani, W. Metzner, R. Wessel, C. Koch, "From stimulus encoding to feature extraction in weakly electric fish", *Nature,* 384 (6609): 564-567, 1996.

[6] R. de Ruyter van Steveninck and W. Bialek, "Real-Time Performance of a Movement-Sensitive Neuron in the Blowfly Visual System: Coding and Information Transfer in Short Spike Sequences", *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 234 (1277): 379-414, 1988.

[7] C. K. Machens *et al*., "Single auditory neurons rapidly discriminate conspecific communication signals", *Nature Neuroscience,* 6(4): 341-342, 2003.

[8] B. D. Wright, K. Sen, W. Bialek, and A. J. Doupe, "Spike timing and the coding of naturalistic sounds in a central auditory area of songbirds", In *Advances in Neural Information Processing Systems 14*, 309-316, MIT Press, Cambridge MA, 2002.

[9] A. Rokem *et al*., "Spike-Timing Precision Underlies the Coding Efficiency of Auditory Receptor Neurons", *J. Neurophysiol.*, 95:2541-2552, 2006.

[10] W. Kistler, W. Gerstner, and J. van Hemmen, "Reduction of Hodgkin-Huxley equations to a single-variable threshold model", *Neural Computation*, 9:1015-1045, 1997.

[11] W. Gerstner and W. Kistler, *Spiking neuron models single neurons, populations, plasticity*, Cambridge University Press, Cambridge, 2002.

[12] D. Floreano and C. Mattiusi, "Evolution of spiking neural controllers for autonomous vision-based robots", In *Evolutionary Robotics IV*, Springer-Verlag, Berlin, 2001.