Department of Information Technology

technical report NUIG-IT-220202

Learning in Artificial Life Societies

D. Curran (NUI, Galway)
C. O'Riordan (NUI, Galway)

Learning in Artificial Life Societies

Dara Curran and Colm O' Riordan,
Dept. of Information Technology,
National University of Ireland, Galway.

February 22, 2002

Abstract

The fields of neural networks and genetic algorithms have both presented alternative paradigms for modelling learning. In the field of neural networks, the brain is modelled in a simplified manner as a network of interconnected nodes; learning is achieved by modifying the weights on the edges. In genetic algorithms solutions are evolved across a population of chromosomes. Learning occurs at a population level whereas in a neural network learning occurs in that network in isolation.

In nature, there is much evidence that these processes neither occur nor operate in isolation (e.g. a learned trait may become genetically assimilated). Many theories regarding the interaction between evolution and life-time learning have been proposed including those of Lamarck, Darwin and Baldwin.

In this paper, we describe the design and development of a simulator which allows the evolution of populations of creatures all of which are capable of learning at the level of an individual correspondingly to life-time learning and population based learning.

We discuss the results of a series of simulations wherein these different types of learning can be used, either in isolation or together. We also investigate their robustness in the face of changing environments.

The work consolidates some existing findings in the field and also suggests that the combination of life-time and population based learning may lead to the development of a paradigm suitable for the development of systems capable of evolving solutions acting in changeable environments.

1 Introduction

The artificial life simulator was developed so that meaningful experiments with artificial life societies comprising a population of neural networks could be performed in such a way that allowed population, individual and combined learning.

At an individual level, each component of a population is a neural network[4],[5] which is capable of responding and reacting to stimuli react. In the case of the simulator, these stimuli were in the form of simple bit patterns which were introduced to the input layer of each network.

Once an output is calculated, it is collected from the output layer of each network and compared to the desired result. Using back-propagation, each network is capable of 'learning' responses to each stimuli.

At a population level, the simulator uses genetic algorithms[14],[2],[3] to increase a population's fitness by selecting the creatures which have performed well and 'mating' their gene codes. The gene codes themselves represent a binary encoded version of each creature's neural network. In this way, it is possible to blend neural networks together in order

to achieve a higher degrees of population fitness.

It was found that by combining both the neural networks' performance and the genetic algorithm, not only was it possible to out-perform each mechanism, but that this also proved to be much more robust in to environmental changes.

In subsequent sections, we will examine the simulator in more detail and discuss some of the results which were obtained during experiments.

2 The Artificial Life Simulator

2.1 Environmental Setting

The environment consists of a number of stimuli that are given to each creature. The stimuli consist of bit patterns representing food and poison. The way the creature responds to the bit pattern determines whether it has correctly identified it as food or poison. Each creature is assigned an "energy level" at birth, which is altered during its lifetime according to its responses to the given bit patterns. Whenever a creature ingests poison, recognising it incorrectly as food, it is punished by removing some of its life energy. When it ingests food it is rewarded.

The life energy of each creature can be used as an evaluation of the creature's ability, but a second method is also incorporated in the simulator. This function examines the creature's performance in terms of its neural network responses. It takes account of the network's overall error as well as how many of the stimuli it responded correctly. To understand why a second "opinion" is sometimes necessary, consider the following example. A creature c is created with a below average neural network. During its lifetime c loses a lot of its life energy as it attempts to distinguish between the food and poison patterns. Toward the end of its lifetime c finally learns to distinguish between the two. However, by this time it will have used up much of its life energy and given the first function alone, would probably not be selected for subsequent generations. Even though the learning took a long time, which may not be immediately useful, a recombination of the creature's gene code with another's may produce a good neural network. Thus the two methods combined provide a solid evaluation of the abilities of each creature.

2.2 Neural Network

The neural network represents the nervous system of each creature in the simulator. The network is central to individual creatures' ability to learn about their environment in order to survive. Many possible architectures were available for consideration and had to be considered according to their merit with respect to complexity, suitability to analysis and to their functionality.

The final choice of network architecture was a fully connected feed-forward network, using the error back-propagation algorithm. It was decided that whilst an auto-associative network may be able to produce some interesting results, its relative complexity may require a prohibitively long development time.

To introduce diversity amongst creatures, the simulator initially assigns randomly the number of layers, the number of nodes per layers and each of the weights. This provides a rich diversity of network architectures, allowing networks of differing shapes and sizes to achieve the same solution.

The neural network component, like the genetic algorithm component can be switched on (and off) to conduct experiments pertaining to the usefulness of life time learning, population learning, and combined learning.

2.3 Genetic Algorithm

The simulator incorporates a genetic algorithm mechanism which allows populations of creatures to

evolve.

The gene code of a creature consists of a string of binary digits. These are encoded in such a way as to represent a neural network. The complex structure of a neural network can be broken down into component parts making it easier to encode.

A neural network consists of nodes and links. For encoding, given that each node in layer n is attached to all nodes in layer $n+1$, it is possible to calculate how many links are between the nodes in each layer. Therefore it is sufficient to encode the number of layers and number of nodes in each layer.

The most important element in a neural network, from a learning perspective, is the value of the edge weights. Upon calculating the number of links per layer then no further information needs to be encoded other than the value of the weights.

At the start of each gene-code, there is a header section which contains the number of layers followed by the number of nodes in each layer. In the remaining portion of the code, each weight is encoded (Figure 1).

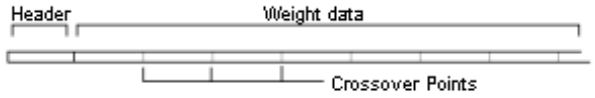

Figure 1: *A sample creature gene code showing header and crossover points*

As weights in a neural network are floating point numbers, a simple method was devised in order to balance the need for sufficient accuracy so that effective learning could be achieved, and the need for a manageable encoding scheme. Following much experimentation, it was determined that 2 significant digits per weight was sufficient for adequate learning without inflating the size of the encoding too much.

Selection is achieved by the standard roulette wheel technique and the whole population is replaced each generation. Population sizes in the simulator are static; while the possibility of dynamics populations, where some genes die out while others live to see their offspring evolve, sounds very appealing, the results would be more difficult to analyse and so was not implemented in the simulator.

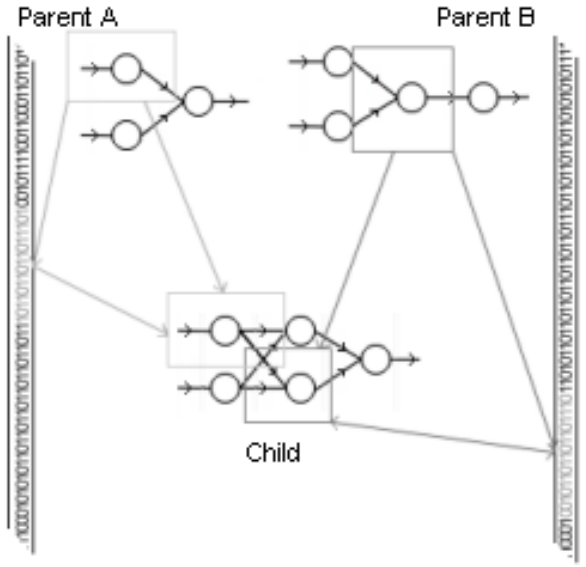

Figure 2: *Sample crossover showing the child gene containing neural network portions of both parents*

One-point crossover, as illustrated in the genetic algorithms section, is used in the simulator to recombine parent gene-codes. However, it is important that the crossover process do not interfere with the developed encoding technique. To ensure that the crossover process does not interfere with the developed encoding technique, it is necessary to take into account the header section and the number of bits required for each encoded weight. Intervals of n bits can then be placed on the gene code and crossover is made to occur in one of these points. The resulting strings will contain information from both parent strings, meaning that different portions of the child's neural network will correspond to different portions of the parent networks (Figure 2). A set mutation rate is also used by the simulator to determine the

chance of a bit in the gene code being flipped.

2.4 Changing Environments

A useful property of the artificial life simulator, for experimentation, is the ability to alter the environment during an experiment. This allows the examination of the creatures' reactions and ability to recover from such changes.

In particular, it is useful in outlining the benefits of life time learning, population learning and the two combined. In order for this to work, it is simply a case of altering the appearance of food or poison during the course of the simulation. This can be done quite readily by specifying an alternate training and testing set to use some specified generation.

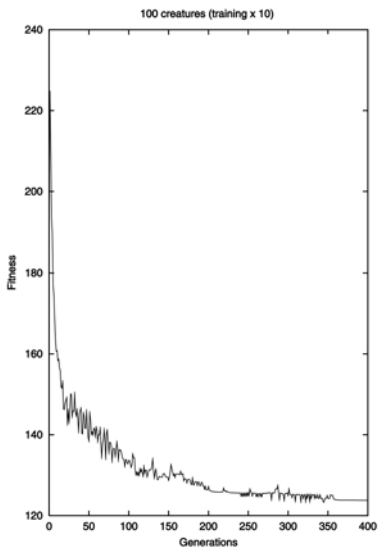

Figure 3: *Individual learning for the XOR problem*

3 Results

The experiments carried out involved a population of 100 creatures evolving for 400 generations. These were set up to identify the relative benefits or shortcomings of individual, population and combined learning when confronted with problems such as the XOR and OR. The creatures were allowed to learn (individual and combined learning), and each was given 10 iterations of training on randomly generated food and poison patterns. In the following sections, the results for each experiments will be outlined.

3.1 Individual Learning

The individual learning experiments, wherein creatures are not subjected to evolution controlled by a genetic algorithms, take a slightly different view of the commonly accepted evolutionary process. As recombination and mutation are not allowed, the process of mating does not yield any new genetic material. This means that if the experiments were set up to produce children identical to the parents, and the neural networks reset according to their gene code each generation, the results would be simply the same for each generation, which is not very useful.

Instead, the Lamarckian theory of evolution is adopted for the purposes of these experiments; rather than resetting the neural networks of the children at each generation, the neural networks are left intact and allowed to carry on into the next generation. It is in fact as though the same creatures were continually present in the population, never dying nor being replaced.

This is implemented by omitting any mating of the creatures. The results are shown for the XOR problem in figure 3 and for the OR problem in figure 4.

The XOR problem is one which cannot be solved easily, as we can see from figure 3. The graph is a simple downward sloping curve, eventually hitting rock bottom fitness at the 120 mark, even though it started at a fairly average 225.

This is due to the fact that, although some creatures will exist that are capable of recognising food, many are not capable. As the training continues,

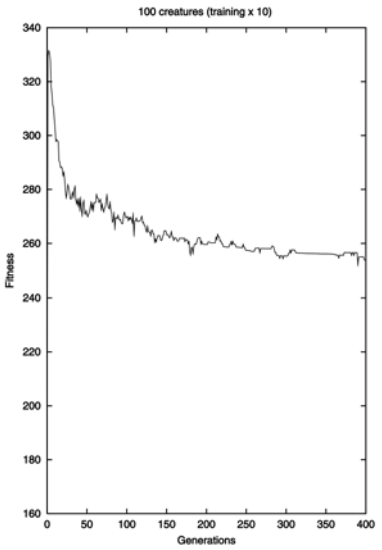

Figure 4: *Individual learning for the OR problem*

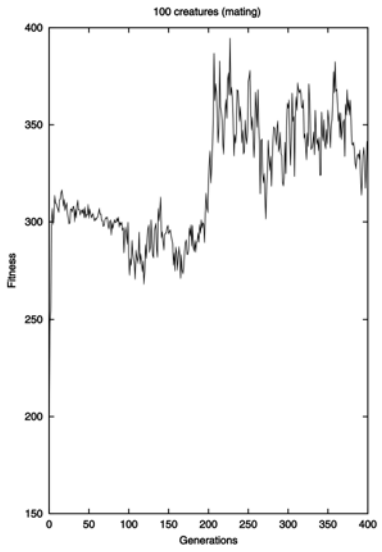

Figure 5: *Population learning for the XOR problem*

these creatures actually become worse at recognising the food, and the average fitness deteriorates. This is because the network's architecture may be simply unsuitable for solving the XOR problem and further training does not rectify, but in fact worsens the condition of the network outputs. The graph for the OR problem (figure 4) shows the same downward sloping curve, although at a higher average fitness, and demonstrates a similar situation.

It is rather clear from these experiments that this type of quasi-Lamarckian evolution is not really suitable for such problems. Individual learning on its own does not seem to lead to the same kind of results as obtained in our subsequent experiments.

3.2 Population Learning

Population learning, implemented using genetic algorithms, is carried out by generating a population of creatures and allowing them to mate over a number of generations. None of the creatures are trained at any time, instead they are tested to determine their fitness with relation to the testing set. Thus, the "nervous system" of each creature remains static throughout its lifetime and the only changes from generation to generation are genetic.

The results for the harder learning problem, XOR, can be seen in figure 5. There is an immediate peak from a low fitness to an average fitness, due to the immediate elimination of obviously poor creatures. The fitness stays around the 300 mark until at around the 200^{th} generation there is a sudden surge in fitness. This is most likely due to the genetic algorithm having "discovered" a gene, or a number of genes which perform very well.

It is clear from the figure that population learning is a powerful tool in the search for an optimal population. Average fitness in any of the experiments rarely exceed the 450 mark so an average fitness of around 400 should be considered very good.

The results for the OR problem (Figure 6) are rather similar, although the fitness does seem to initially drop and then recover rapidly.

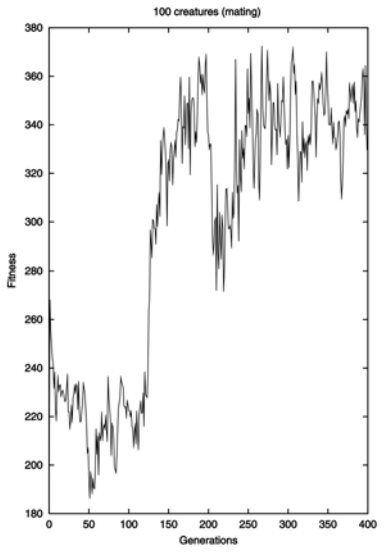

Figure 6: *Population learning for the OR problem*

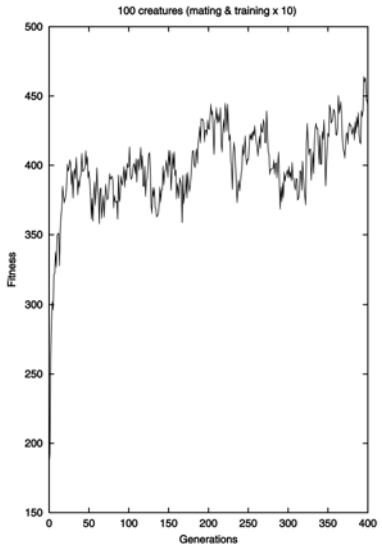

Figure 7: *Combined learning for the XOR problem*

3.3 Combined Learning

The next set of experiments performed were concerned with the effect of combining both population based learning given by the genetic algorithm component of the simulator and the individual learning component.

Each individual creature in the population is "born" with a gene code which remains static throughout its lifetime. As the creature is trained, the neural network will be altered as will its perception of what represents food and poison. This change in the neural network is not re-encoded into the gene code of the creature or its offspring. During reproduction, two creatures create two new child creatures which will contain segments of each of the parent's gene code and therefore segments of each parent's neural network.

The graph in figure 7 shows a sharp increase from an initial fitness of about 190 to 400 for the XOR problem. The fact that results clearly outperform both the population and individual learning experiments described previously deserves some attention.

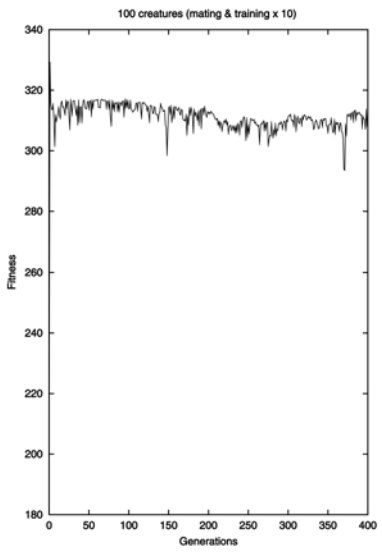

Figure 8: *Combined learning for the OR problem*

It is interesting to note that in these experiments, the creatures' performance is superior to that achieved in the experiments using only population learning results, without re-encoding what they have learned during their lifetime into genetic code. Rather, it seems that those creatures which successfully learned had a higher fitness than those which did not, and so were chosen as parents. After about the 20^{th} generation, most of the existing population demonstrates this predisposed ability to distinguish between food and poison. We see evidence of the *Baldwin effect*[15] (where learned traits have become genetically assimilated into the population).

The graph for the OR results (figure 8) shows a different situation. The population fitness seems almost static at around 310 for the duration of the experiment. This provides an example of the hiding or shielding effect, where the problem at hand is trivial enough to be learned quickly by most members of the population, thereby hiding genetic differences and making selection more difficult for the genetic algorithm (known as the hiding or shielding effect[10]).

3.4 Changing Environments

These experiments were designed to determine the robustness of the process of individual learning, population learning and the two combined. The population begins with a standard training and testing set for 250 generations. Having completed these generations, the training and testing sets are reversed, i.e., that which used to represent food now represents poison and vice versa. This equates to a catastrophic change in the environment. The results are described for each of the three cases for both harder and easier problems in turn.

3.4.1 Population Learning

The results for the XOR problem are illustrated in figure 9. The sudden drop in fitness at the 250^{th} generation is dramatic, dropping from an average 300 to 50. However, the population appears to start recovering and reaches 150 by the time the experiment ends. The huge drop is to be expected, because the population learning process is too slow to adapt to such a sudden change. In fact, if the simulator supported dynamic populations, there probably would not have been many creatures left at the 250^{th} generation.

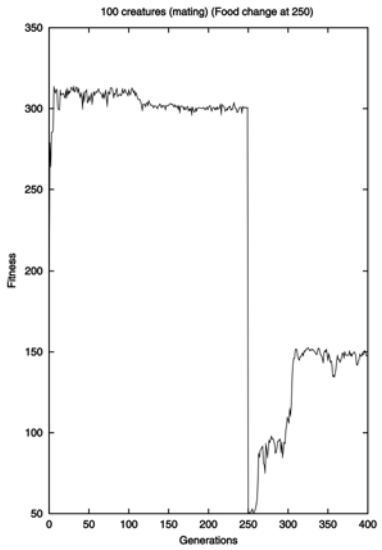

Figure 9: *Population learning for the XOR problem in a changing environment*

The population is able to recover to a degree and may continue to recover during subsequent successive generations. The recovery speed is quite similar to the beginning of the experiment with population fitness in both cases jumping by about 100. The population learning is quite weak at managing sudden change, but will eventually recover from such trauma. The situation is almost the same for the OR problem (figure 10) where the graph exhibits much the same fluctuations, sudden drop and slight recovery.

3.4.2 Individual Learning

The XOR problem results in figure 11 show a similar tendency toward a downward curve like the previous

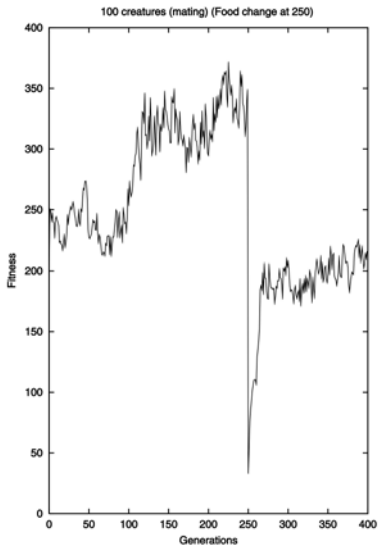

Figure 10: *Population learning for the OR problem in a changing environment*

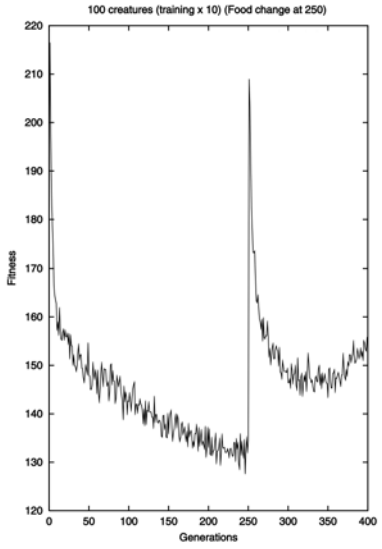

Figure 11: *Individual learning for the XOR problem in a changing environment*

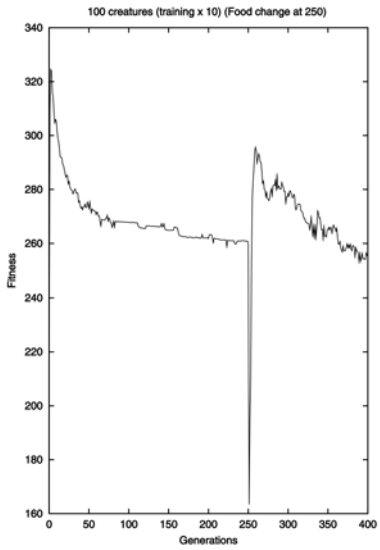

Figure 12: *Individual learning for the OR problem in a changing environment*

XOR experiment with individual learning. In this case, the sudden change in food and poison patterns seem to have actually been initially beneficial to the population. The sudden jump from a dwindling 130 to an average 210 seemed promising, but the population quickly fell back to nearly its mediocre level. However, the boost was not completely lost, as for the first time, the average fitness of the population seems to be on the increase toward the end of the experiment.

Had the change been the other way around, the population would have quickly succumbed. However, the boost incurred from the change allowed the population to begin to improve their fitness. It can be deduced that individual learning in isolation is certainly not robust for more difficult problems and the effects any change will have are entirely dependent on the population in existence.

The situation for the OR problem illustrates how a full recovery may be made with an easy problem (figure 12). The graph begins by exhibiting the familiar downward sloping curve. The abrupt drop

corresponds to the change in patterns. However, instead of remaining at a low fitness, the population quickly recovers.

This is best explained by the fact that the training and testing sets for both the initial and final stages of the experiment both represent very easy problems. The neural networks take no time at all in reaching an acceptable fitness given both the first and second food and poison patterns.

3.4.3 Combined Learning

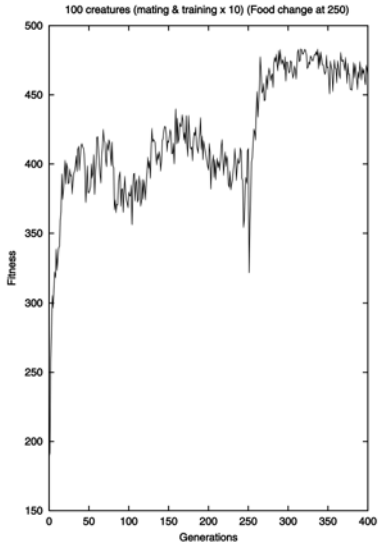

Figure 13: *Combined learning for the XOR problem in a changing environment*

The final experiment involved both individual and population learning in a changing environment. The results for the XOR problem illustrated in figure 13 show that even though the population was severely affected by the sudden change in food and poison patterns, within a few generations it had recovered and increased its fitness to an even higher level than it had before the change occurred.

The graph for the OR problem (figure 14) shows a near flat fitness of about 310. There is hardly a perceptible sign of the change in pattern occurring as the fitness remains almost entirely stable throughout. The graph is similar to the previous experiment involving combined learning and also to the individual learning experiment above. As the new training set is also trivial to learn, the training element of the population does not even require a generation to recover from a change in patterns.

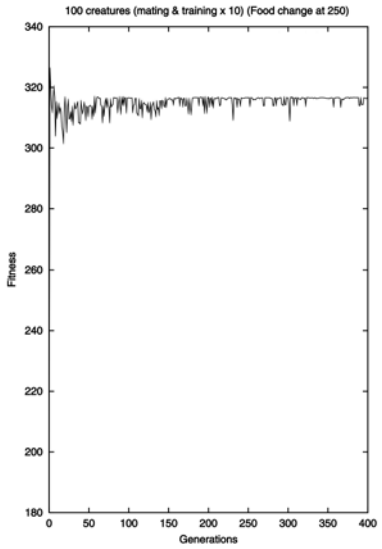

Figure 14: *Combined learning for the OR problem in a changing environment*

Such results outline the robustness of the combination of both individual and population learning. These findings echo related results obtained; the interested reader is directed to[6],[8] and [9] for related work. The population fitness did not drop to disastrous levels because the individual learning factor allowed the creatures to at least learn *something* about the new patterns. The ones which were found to have learned better than others were then given the opportunity to pass on their learning ability, after which the population recovered.

4 Conclusion and Possible Future Developments

The construction of the simulator has brought together many aspects of the artificial life world, but could be expanded to include features which would give the ability to set up more intricate experiments. However, the experimental results which the simulator allowed to generate are interesting.

The distinction between the Darwinian and Lamarckian ideas of evolution was able to be played out in the simulator leading to some interesting conclusions. While the two, represented by population and individual learning respectively, did not fare particularly well in isolation, when they were combined, the results outperformed either learning mechanism in isolation.

Behaviours exhibited by the populations demonstrated both the Baldwin effect and the hiding effect.

Extra functionality could be added to the simulator to allow more sophisticated experiments. With respect to neural networks, it would be interesting to implement auto-associative neural nets as well as associative. It would be interesting to see how each would respond individually and in the mixed in the same population.

With respect to the genetic algorithm, only 1 point crossover was implemented, leaving room for additional work which might increase the performance of the genetic algorithm component. Also, the encoding mechanism, while useful in this implementation could be improved to be more efficient.

The simulator could be conceivably altered to evolve neural networks capable of solving more complex problems than OR or XOR. In most of the experiments, while the population fitness might not have been as high as hoped, the population was always able to correctly solve these two problems, suggesting that the idea of the simulator as a problem solving platform may be useful.

References

[1] Darrell Whitley, *A Genetic Algorithm Tutorial*, Computer Science Dept., Colorado State University, 1993

[2] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Inc., 1989

[3] Melanie Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts, 1996

[4] Igor Aleksander and Helen Morton, *An Introduction to Neural Computing*, International Thomson Computer Press, 1995

[5] Mohamad H. Hassoun, *Fundamentals of Artificial Neural Networks*, MIT Press, Cambridge, Massachusetts, 1995

[6] Takahiro Sasaki and Mario Tokoro, *Adaptation toward Changing Environments: Why Darwinian in Nature ?*, Department of Computer Science, Keio University, Japan

[7] Federico Cecconi, Filippo Menczer and Richard K. Belew, *Maturation and the Evolution of Imitative Learning in Artificial Intelligence*, Institute of Psychology, National Research Council, Rome, Italy

[8] Stefano Nolfi and Domenico Parisi, *Learning to Adapt to Changing Environments in Evolving Neural Networks*, Institute of Psychology, National Research Council, Rome, Italy

[9] Orazio Miglino, Stefano Nolfi and Domenico Parisi, *Discontinuity in Evolution: How Different Levels of Organization Imply Pre-Adaptation*, Department of Psychology, University of Palermo, Italy

[10] Giles Mayley, *Guiding or Hiding: Explorations into the Effects of Learning on the Rate of Evolution*, School of Cognitive and Computing Sciences, University of Sussex, Brighton, England

[11] Melanie Mitchell, Stephanie Forrest, *Genetic Algorithms and Artificial Life*, Santa Fe Institute, Santa Fe

[12] Melanie Mitchell, Stephanie Forrest, *Genetic Algorithms and Artificial Life*, Santa Fe Institute, Santa Fe

[13] Melanie Mitchell, Stephanie Forrest, *Genetic Algorithms and Artificial Life*, Santa Fe Institute, Santa Fe

[14] John Holland, *Adaptation in natural and artificial systems*, Ann Arbor, 1975

[15] JM Baldwin JM, *A new factor in evolution*, 1896, American Naturalist, volume = 30, pp 441-451.