2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Augmenting the NEAT Algorithm to Improve its Temporal Processing Capabilities

Pilar Caamaño, Francisco Bellas, and Richard J. Duro

Abstract—**This paper is concerned with the incorporation of new time processing capacities to the Neuroevolution of Augmenting Topologies (NEAT) algorithm. This algorithm is quite popular within the robotics community for the production of trained neural networks without having to determine a priori their size and topology. However, and even though the algorithm can address temporal processing issues through its capacity of establishing feedback synaptic connections, that is, through recurrences, there are still instances where more precise time processing may go beyond its limits. In order to address these cases, in this paper we describe a new implementation of the NEAT algorithm where trainable synaptic time delays are incorporated into its toolbox. This approach is shown to improve the behavior of neural networks obtained using NEAT in many instances. Here, we provide some of these results using a series of typical complex time processing tasks related to chaotic time series modeling and consider an example of the integration of this new approach within a robotic cognitive architecture.**

I. Introduction

Within the robotics community there has always been a need for algorithms that allow to automatically obtain neural network based structures that permit performing different control and perception tasks. In fact, this need has become even more pressing as more and more work has been devoted to obtaining Cognitive Architectures (CAs). These architectures are the computational implementation of cognitive models [1], and as such, constitute the substrate of functionalities like perception, attention, action selection, learning, reasoning, etc. In the last few years the subfield of Cognitive Developmental Robotics (CDR) [2] has become a source of many of the most popular CAs. The bases of CDR were articulated by Weng in [3] who indicated that "a developmental architecture requires not only a specification of processors and their interconnections, but also their online, incremental, automatic generation from real-time experience". That is, the control structure of the robot must be obtained autonomously by the robot through interaction with its environment and not produced through some explicit prior design.

Thus, when considering neural network based controllers, a need for algorithms that can produce them with the appropriate size and structure for a given set of input-output relationships is evident. In this line, Neuroevolution of Augmenting Topologies (NEAT) [4][5] has been a very successful algorithm for evolving Artificial Neural Networks (ANNs) that adapt their structure and processing to the task that is required from them. This evolutionary algorithm has been applied successfully for obtaining the weights and structure of ANNs in different domains going from data classification [6][7] to evolutionary robotic design [8]. However, its main application field has been that of learning in dynamic domains, like video games [9][10][11] or vehicle crash simulation [12]. Its operation is based on the use of history markers in genes to promote crossover between similar topologies. Thus, species or niches in the population are preserved by avoiding reproduction between historically different individuals. Moreover, NEAT starts with simple feed-forward ANNs that contain only input and output neurons and it incrementally increases their complexity through structural mutation operators, the add connection mutation and the add node mutation [13]. This way, a designer does not need to predetermine the architecture and number of nodes of the ANN needed for a given task or function.

In the domain of ANN based cognitive architectures, one of the primary functions of many of the networks developed is to produce models of reality that can be used by the robot's cognitive architecture when deciding on its actions and strategies. These models often have to deal with temporal relationships and, consequently, the ANNs produced must be able to somehow model these temporal aspects in an intrinsic manner. NEAT is able to manage time dependent phenomena through recurrent or feedback connections inserted between neurons using its add connection mutation operator. Therefore, it intrinsically supports the generation and training of classical Recurrent Neural Networks. However, classical Recurrent Neural Networks (RNN) present several drawbacks when dealing with problems that require precise timing [14], especially when modeling the underlying structure of complex time series, and different approaches have been developed to address them [14][15]. It turns out that one of the most popular implies introducing controllable time delays in the feedforward or feedback synapses of the networks, leading to the concept of Time Delay Recurrent Neural Networks (TDRNN) [16][17][18].

The underlying theoretical results in signal processing that support their use are based on the embedding theorem [19][20] which states that an unambiguous model of any dynamical system characterized by a measured signal can be obtained by embedding this signal in a higher dimensional space of dimension D. This embedding can be achieved by

All authors are with the Integrated Group for Engineering Research, University of A Coruña, Ferrol, Spain (phone: +34981337400, e-mail: {pcsobrino, fran, richard}@ udc.es).

This work was partially funded by the Spanish MICINN and European Regional Development Funds through project TIN2011-28753-C02-01.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

taking D samples of the signal spaced by an amount τ. In signal processing terminology, the dynamic reconstruction of the signal is possible this way.

From this point of view, the only problem that remains is how to obtain D and τ, and most time delay based ANNs assume that by appropriately training a set of synaptic delays in the networks, one automatically obtains these two parameters in an intrinsic manner. That is, these delays can be taken as a representation of the different lengths of these connections and different synaptic lengths imply different amounts of time for the signals to traverse them.

In fact, lower dimensional embedding spaces could be used if the samples were not evenly spaced in time and, thus, by considering an uneven distribution of delays, many dynamic processes could be modeled unambiguously using a smaller number of signal points [21].

Most current TDRNN training algorithms do not provide for these networks to be grown and adapt their topology and weights to the problems they try to solve as the NEAT algorithm does. Consequently, in this paper, we argue that by adding the capability of incorporating and managing delays to NEAT, better signal modelling ANNs can be obtained. In fact, we discuss how this capability may be added to NEAT and present some experimental results showing how this improves the networks obtained.

The rest of the paper is organized as follows. Section 2 presents the τ-NEAT algorithm, which is our proposal for adding the capability of introducing time delays to NEAT. Section 3 shows the results of some experiments carried out over a series of chaotic time series often used in order to test temporal modeling structures. In Section 4 we apply τ-NEAT to a real robotic cognitive architecture. Finally, section 5 presents a series of conclusions.

II. Modification of the NEAT algorihtm

As commented above, this section is devoted to the presentation of how the NEAT algorithm has been extended in order to be able to manage synaptic delays. This extension of the NEAT algorithm is called τ-NEAT.

τ-NEAT is, basically, a neuroevolutionary algorithm for growing neural networks that may include recurrent connections and synaptic delays. Fig. 1 displays the structure of a general or prototypic neural network that τ-NEAT may obtain. For every synapse between neurons i and j, this network includes a synaptic delay τ_{ij}, in addition to the synaptic weight w_{ij}. This time delay is modeled by means of a FIFO buffer containing the last τ_{ij} input values to that synapsis.

Concerning the operation of the algorithm, it follows the same basic structure as NEAT (described in [4]) with some slight modifications. On the one hand, and as indicated before, the synaptic delays have been included in the NEAT chromosome and their value determines the size of the buffer and, consequently, the length of the synaptic connection and or the time the signal needs to traverse it. They are handled much in the same way as if they were synaptic weights in the original implementation with the limitation that they are integer numbers. Thus, in terms of evolution, the τ-NEAT approach works in a very similar manner to the original NEAT algorithm with the exception of the operators that are necessary to evolve and manage the τ value. That is, it was necessary to extend the synaptic connection to deal with the delays and, as a consequence, a new mutation operator that modifies the τ values each generation was added.

Obviously, the delays affect the inputs to each neuron, that is, the time at which each output of the previous neuron reaches the target neuron. The inputs to the target neuron are now dependent on the τ values which determine the length in time steps of a buffer these values have to traverse to reach the targets. The new input is stored in the buffer, which works as a FIFO with a maximum capacity of τ_{ij}.

Another consideration to take into account when working with this new model is that its behavior depends on values in previous temporal instants. Due to this fact, the outputs of τ-NEAT cannot be considered as valid until every buffer is filled. This is, in order to produce significant outputs, every neuron in the network must have a complete set of inputs and, as there are synaptic delays throughout the network, it takes a given number of iterations for all the neurons to have relevant values in all of their inputs, that is, to fill the synaptic buffers.

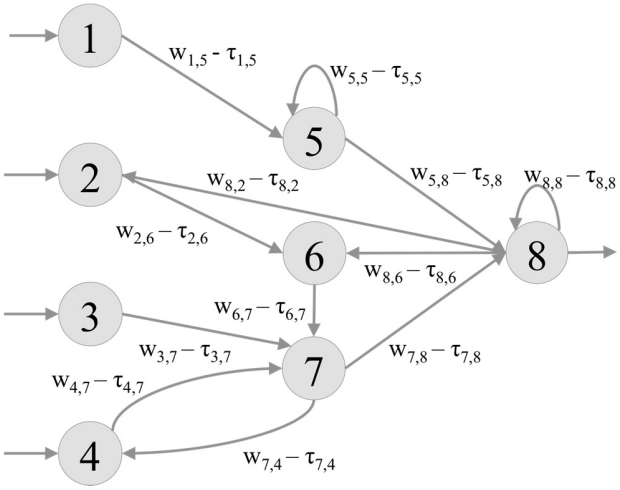

Fig 1. Structure of an example τ-NEAT neural network

III. Some Tests

This section is devoted to the presentation of the results of a series of tests over benchmark functions that were carried out in order to compare the performance of the τ-NEAT algorithm to that of the standard NEAT algorithm when modeling temporal series. As indicated before, the objective here is to produce ANNs that are able to model temporal processes learning from a signal produced by measuring the corresponding dynamic system. Thus, we have chosen a set of well-known benchmark signals used in many signal

processing tasks, the Logistic map and the Mackey Glass time series. These series are chaotic, at least for some parameter intervals, and we have chosen these intervals for the tests. The Logistic map is a very nice signal for benchmarking as its state space can be plotted in two and three dimensions and, consequently, it is easy to see if the networks have really captured the underlying system from the signal they were presented with. On the other hand, the Mackey Glass time series is much more complex to model and visualize. In most implementations, in order to disambiguate a value it is usually necessary to consider 17 or more previous values. This makes it more interesting for testing signal modelers, at the expense of not being able to plot its state space and of having to analyze the performance of different approaches over the observable signal. In every case, we have compared the results of the τ-NEAT algorithm to those provided by the original NEAT algorithm as presented in [4].

A. Logistic Map

The logistic map chaotic time series is defined by the following equation:

$$x_{n+1} = r x_n (1 - x_n)$$

Where x_n is the current value of the signal and x_{n+1} the next one. Its behavior depends on the value of r, which is a positive number that represents a combined rate for reproduction and starvation in population dynamics. However, as we are only interested in having a chaotic signal, we have fixed this value to $r=4$ and the starting value to $x_0=0.1$. The phase diagram of the logistic map is displayed Fig. 2. The left graph provides a two-dimensional representation showing its quadratic behavior while the three-dimensional representation is shown in the right graph.

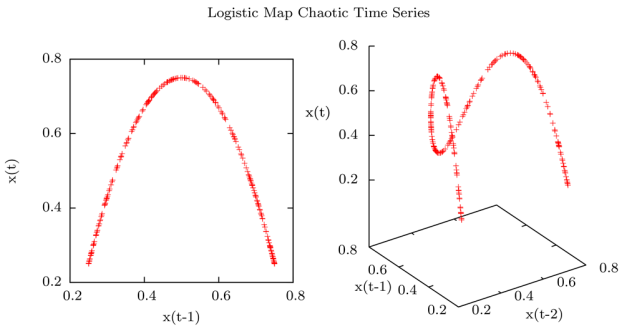

Fig 2. Logistic map time series phase diagram. Two-dimensional representation (left). Three-dimensional representation (right).

The experiment seeks to establish a comparison between the performance of the τ-NEAT and standard NEAT algorithms. Consequently, the two models were tested:

- The original NEAT model that uses recurrences in its networks in order to model temporal processes.
- The τ-NEAT approach presented in this work which, in addition to the recurrences used in the original NEAT model, includes time delays in its synapses.

The common parameters used in the configuration of these models are displayed in Table 1. Those are the values recommended by the NEAT authors, except in the case of the mutation operator.

As modeling chaotic time series requires a good precision level and due to the influence on result precision of the mutation operators, here we have resorted to the Michalewicz non-uniform mutation operator [22] in order to improve the precision of the results. This mutation operator automatically adjusts the mutation step size during the evolutionary process. It makes it large at the beginning of the evolutionary process, allowing the algorithm to extensively explore the search space, and then it decreases it with the number of generations. In the last generations of the evolutionary process, the mutation step size takes low values and, consequently, only small changes of the weights take place, thus permitting a more precise honing of the final values.

TABLE I

COMMON CONFIGURATION PARAMETERS

Generations	1000
Population size	500
Topology Mutation	Classic
Add Connection Rate	0.1
Remove Connection Rate	0.01
Remove Connection Max Weight	5.0
Add Neuron Rate	0.1
Prune Mutation Rate	1.0
Weight Mutation	Michalewicz Non Uniform Mutation
Weight Mutation B parameter	5.0
Weight Range	[-10.0:10.0]
Survival Rate	0.1
Elitism	True
Selection operator	Roulette
Elitism species minimum size	1

The fitness value used to compare the individuals of the population was the MSE error in a minimization process. These errors are only taken into account from the moment the buffers of the time-delayed synapses are filled. This is done because it would make no sense to consider errors in instants when some of the neurons may not have any input, as the corresponding buffer position has not yet been filled.

Finally, the number of recurrences and the maximum buffer size (maximum time delay) used in the time-delayed synapses are presented in Table 2.

TABLE II

SPECIFIC CONFIGURATION PARAMETERS

	NEAT	τ-NEAT
Recurrence Type	Best Guess	Best Guess
Recurrence Cycles	5	2
Time-Delayed Synapsis buffer size	0	2

With these configurations, the two models were evolved. The results obtained after 50 independent evolutions are displayed in Fig. 3, Fig. 4 and Fig. 5. The first one represents the evolution of the average MSE error. Whereas Fig. 4 and Fig. 5 show the phase diagrams after applying the best individuals obtained in each case to the prediction of the logistic map. The left graphs correspond to the NEAT algorithm and the right ones to τ-NEAT.

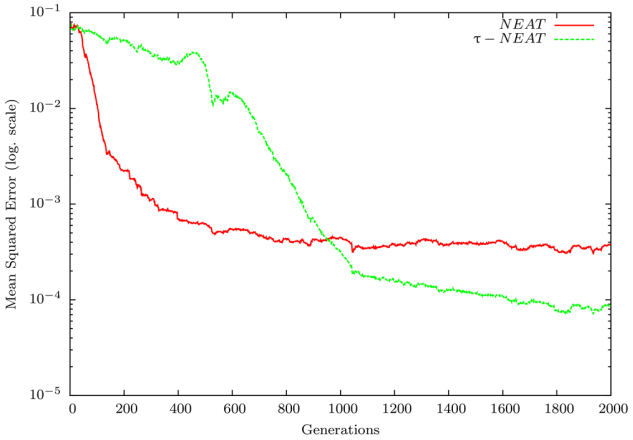

Fig. 3. Average MSE error for 50 independent evolutions of the logistic map prediction experiment.

Each of the graphs in figs. 4 and 5 display three data sets. The small blue dots represent the original time series to be predicted. The red crosses correspond to the outputs of the ANN when the inputs used by the network are the values of the original time series, that is, for a prediction one instant into the future. This test permits analyzing how well the resulting ANN approximates the time series that it has to predict. Finally, the green xs have to do with the time series obtained by the ANN when the output in time t is used as input for obtaining the new output in time t+1 in a multistep prediction fashion, that is, when we are using the network as an autonomous signal generator. This permits ascertaining how well the network has captured the underlying dynamic system.

As the results show, the model that best fits the Logistic Map time series is the one obtained using τ-NEAT, i.e., the one that uses recurrences and time-delayed synapsis. This is the case both for one-step ahead prediction or when using a multistep approach as a signal generator. From the error plots of Fig. 3, this can also become clear, as τ-NEAT achieves an error value that is more than one order of magnitude better than NEAT in the allotted number of generations. The network that was obtained at the end of the process is made up of 13 hidden neurons and 27 connections.

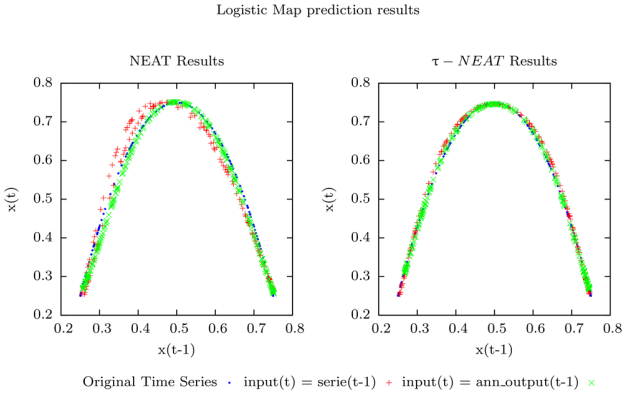

Fig. 4. Logistic map two dimensional phase diagram results. From left to right, results obtained by the NEAT model, and by τ-NEAT.

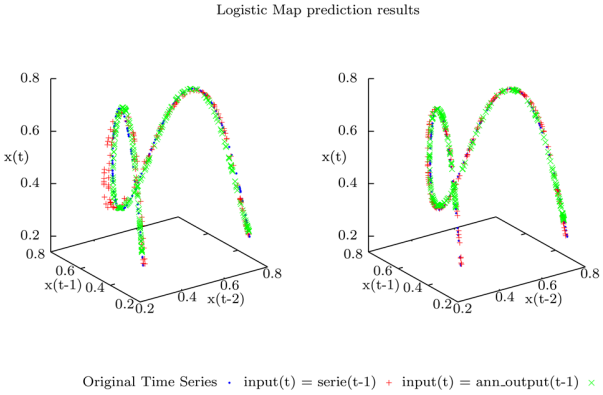

Fig. 5. Logistic map three dimensional phase diagram results. From left to right, results obtained by the NEAT model, and by τ-NEAT.

B. *Mackey-Glass Time Series*

As a second comparison, were carried out an experiment using the Mackey-Glass delay differential equation as the underlying dynamical system. This equation is given by:

$$\frac{dx(t)}{dt} = -bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)^{10}}$$

which is a rather complex chaotic time series for values of τ of 17 or more. It presents non-linearities, limit cycle oscillations and aperiodic waveforms. For these tests the values for τ, a, b and x_0 were set to 17, 0.2, 0.1 and 0.8, respectively, and the step size Δt was set to 1. Moreover, we have used the same algorithm configuration as in the previous case, except for the maximum delay or buffer size, which was set to 18.

Fig. 6 displays the evolution of the average MSE error after 50 independent evolutions. Again, τ-NEAT outperforms NEAT in terms of the error level produced by the final ANN. In terms of the behavior of the best networks obtained by the two approaches, Fig. 7 presents the results of applying them to this problem. Once more, results of the approaches with time-delayed synapses are more precise, being this more

noticeable in the case of the peaks. The τ-NEAT model obtained in this experiment consists of 11 hidden nodes and 32 connections.

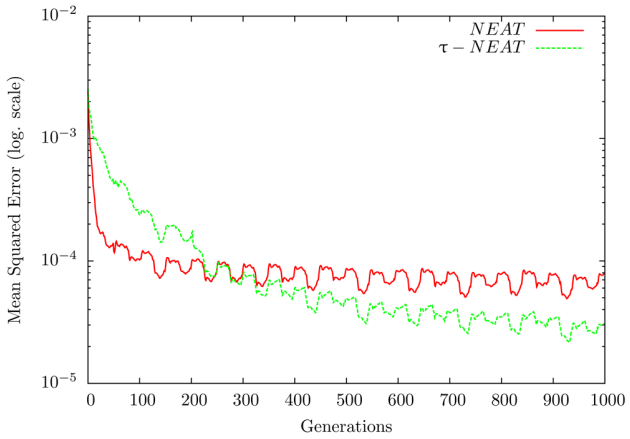

Fig. 6. Average MSE Error evolution for the Mackey-Glass time series prediction experiment.

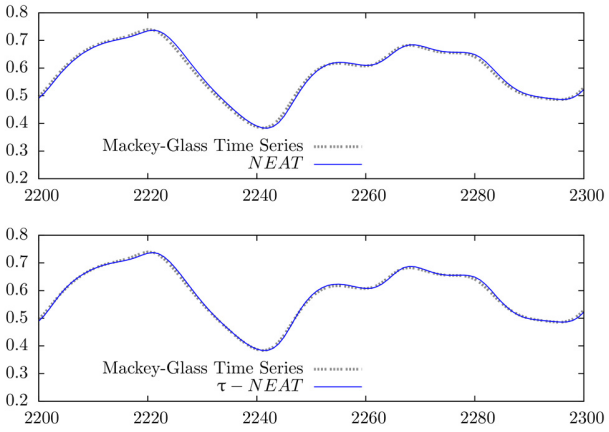

Fig. 7. Mackey-Glass predition results for the three ANN models. From top to bottom, NEAT and τ-NEAT.

IV. Robotic Experiment

A robotic experiment has been implemented to analyze the relevance of using the τ-NEAT algorithm in the Multilevel Darwinist Brain (MDB) cognitive architecture [23]. This experiment involves the robot deciding when to traverse a corridor that is being monitored by a "security guard" that moves across it with different motion patterns. For the robot to cross safely, it must avoid touching the guard. However, as it takes some time to move across the monitored section, the decision to move must come about as a consequence of predicting the motion of the guard several instants in advance. In terms of the MDB, this means creating a model of the guard's motion, that is, producing an ANN that can be used to provide a prediction of when the guard will not be in the robot's path.

Fig. 8 displays six snapshots of the "safe crossing" experiment. We have an Aibo ERS-7 robot and an e-puck robot with a pink ball on its top (guard) that crosses in front of the Aibo. As indicated in the previous paragraph, the objective of the MDB is to learn the models required by the Aibo to advance without running over the guard using, in this case, the τ-NEAT algorithm. The desirable situation is the one shown in the bottom left image while an undesirable one is that displayed in the bottom right image. A schematic overview of this setup can be observed in the top image of Fig. 9. As shown, the robot is placed at a fixed distance from the guard, which performs a continuous and linear movement in front of the robot according to a pre-specified temporal pattern. The Aibo must select the appropriate instant to move and cross without running over the guard. Depending on the temporal pattern followed by the guard, this selection may become very complex requiring a precise temporal modeling to anticipate the guard's position.

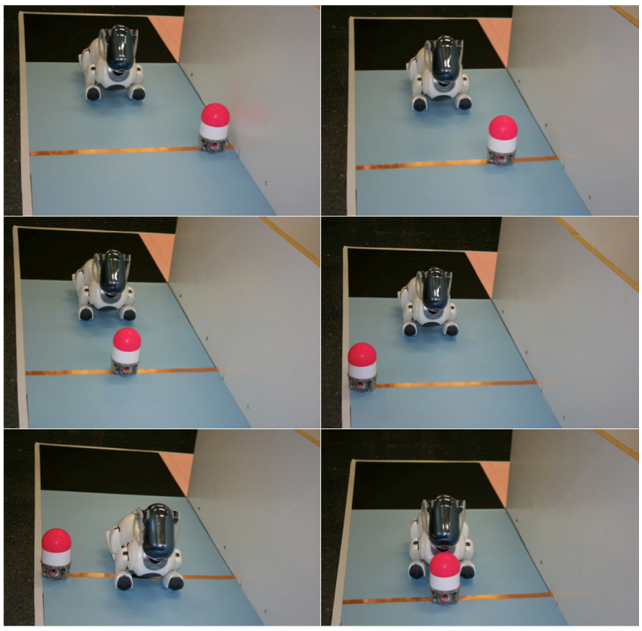

Fig. 8. Snapshots of the Aibo robot "safe crossing" experiment.

In this configuration, the robot has a permanent vision of the guard (pink ball) from its starting position. The Aibo employs its camera, placed in its head, to obtain an estimation of the *distance* and the *angle* to the ball. Specifically, in each iteration, the robot moves its neck from 90º left to 90º right having a complete view of the environment in front of it. If a ball is detected during this neck displacement, the robot centers this ball in the camera image. After that, the distance is calculated as a function of the size of the detected ball and the angle to the ball is the angle of the neck actuator.

The robot can perform two actions: *move forward* or *not move*. If the robot decides to move forward when the guard is crossing the "y" axis, it collides with it and, consequently, the distance and the angle are zero. In any other case, when the robot moves forward, it reaches the origin of coordinates but cannot see the guard, so the distance the sensor returns after processing the image is 3 meters (out of range). Otherwise, the specific values of distance and angle vary in a continuous range from 0 to 1 (*distance*) and from -1 to 1 (*angle*).

We have implemented four different patterns of guard motions to illustrate τ-NEAT's response in different situations. They are precise temporal patterns, so it is assumed that τ-NEAT will perform successfully over them.

For each case, the position of the guard in each iteration can be viewed in the four bottom graphs of Fig. 9 (for example, Function 0 corresponds to a repetitive movement of the guard from [0,0] to [1,0] and back to [0,0]). To be able to address this experiment in the MDB, two models must be considered and learned: a world model and a satisfaction model. Here we will concentrate on the world model.

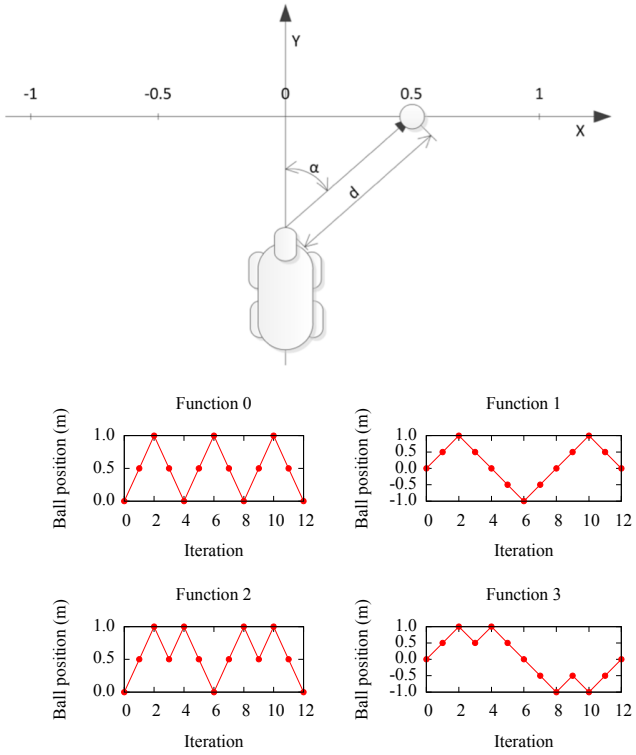

Fig. 9. Top: "Safe crossing" experiment schematic overview. Bottom: Guard position with respect to the AIBO position in the four movement patterns considered.

The world model has three inputs (*distance*, *angle* and *action*) and two outputs (predicted *distance* and *angle*). This model is represented by means of an ANN obtained by the τ-NEAT algorithm.

Fig. 10 displays the evolution of the mean squared error averaged for the two outputs of the world model provided by the NEAT (with recurrent connections but no delays) and τ-NEAT algorithms when the guard follows the four dynamic patterns shown in the bottom plots of Fig. 9. It can be clearly observed that the τ-NEAT algorithm outperforms the original one in all cases, which was the main objective of this experiment. In practical terms, the AIBO robot successfully accomplished the task when the error level was below 1e-03, while in any other case the robot behavior was unstable. As displayed in Fig. 10, such error level was obtained by the τ-NEAT algorithm in all the experiments.

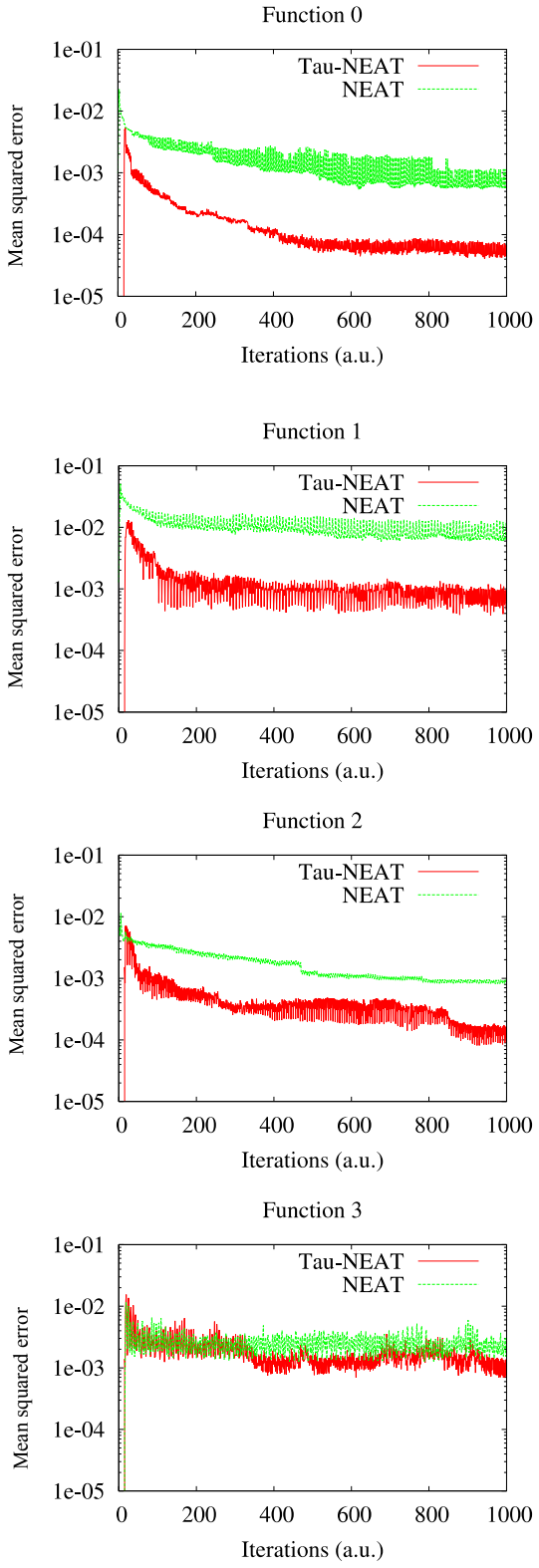

Fig. 10. Evolution of the error provided by NEAT and τ-NEAT.

V. CONCLUSIONS

This paper presents an extension to the NEAT algorithm in order to allow it to produce better signal modelers, which are often required when trying to obtain cognitive architectures for robots. To this end, we have proposed the incorporation to NEAT of the capability of including and managing synaptic delays in the synapses it introduces in its networks, whether feedforward or feedback. The introduction of these terms and capabilities does not change the NEAT algorithm very much and it just requires a few new operators to be able to handle the adaptation of these delays to the problem in hand. This new extension of the algorithm has been called τ-NEAT.

τ-NEAT has been described in the paper and its performance compared favorably to the standard NEAT algorithm over a set of signal modeling cases involving two standard benchmark chaotic time series. In addition, as this work has the objective of producing algorithms that can be used in robot cognitive architectures, a third experiment was carried out where the τ-NEAT algorithm was incorporated to the Multilevel Darwinist Brain (MDB) robotic cognitive architecture for world modeling tasks that require precise temporal processing. In the experiment, τ-NEAT again outperformed the standard NEAT algorithm.

We are now working on more complex robotic tasks and on the integration of τ-NEAT at other levels of the cognitive architecture.

REFERENCES

[1] Byrne, M.D: Cognitive architecture, The humancomputer interaction handbook, Taylor & Francis, vol. 44, no. 1, 97-117 (2003)

[2] Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y.; Ogino, M.; Yoshida, C.: Cognitive Developmental Robotics: A Survey, IEEE Trans. On Autonomous Mental Development, vol. 1, no. 1, 12-34, (2009)

[3] Weng, J: On developmental mental architectures, Neurocomputing, vol. 70, no.13-15, 2303-2323, (2007).

[4] Stanley, K.O., Miikkulainen, R., Evolving neural networks through augmenting topologies. Evolutionary Computation 10 (2), (2002), pp. 99–127.

[5] Stanley, K.O., Miikkulainen, R., Efficient evolution of neural networks topologies. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), (2002), pp. 569-577

[6] Wang, G., Cheng, G., Carr, T.R., The application of improved NeuroEvolution of Augmenting Topologies neural network in Marcellus Shale lithofacies prediction, Computers and Geosciences, 54, (2013), pp. 50-65.

[7] Chen, L., Alahakoon, D., NeuroEvolution of augmenting topologies with learning for data classification, 2nd International Conference on Information and Automation, ICIA 2006, (2006), pp. 367-371

[8] Krčah, P., Towards efficient evolution of morphology and control, GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008, (2008), pp. 287-288.

[9] Stanley, K.O., Bryant, B.D., Miikkulainen, R., Real-time neuroevolution in the NERO video game, IEEE Transactions on Evolutionary Computation, 9 (6), (2005), pp. 653-668

[10] Raffe, W.L., Zambetta, F., Li, X., Neuroevolution of content layout in the PCG: Angry bots video game, 2013 IEEE Congress on Evolutionary Computation, CEC 2013, (2013), pp. 673-680.

[11] Cardamone, L., Loiacono, D., Lanzi, P.L., Evolving competitive car controllers for racing games with neuroevolution, Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference, GECCO-2009, (2009), pp. 1179-1186

[12] Kohl, N., Stanley, K., Miikkulainen, R., Samples, M., Sherony, R., Evolving a real-world vehicle warning system, GECCO 2006 - Genetic and Evolutionary Computation Conference, 2, (2006), pp. 1681-1688.

[13] Stanley, K.O., Miikkulainen, R., Competitive coevolution through evolutionary complexification, Journal of Artificial Intelligence Research, 21, (2004), pp. 63-100.

[14] Gers, F.A., Schraudolph, N., Schmidhuber, J., Learning precise timing with lstm recurrent networks. Journal of Machine Learning Research vol 3 (2003), pp. 115-143

[15] Renart, A., Recurrent networks learn to tell time, Nature Neuroscience 16, (2013), pp. 772–774

[16] Marom E, Saad D, Cohen B., Efficient Training of Recurrent Neural Network with Time Delays, Neural Networks vol 10(1), (1997), pp. 51-59.

[17] Sung-Suk Kim, Time-delay recurrent neural network for temporal correlations and prediction, Neurocomputing Vol 20, Issues 1–3, (1998), pp. 253-263

[18] Boné, R., Crucianu, M. de Beauville, J.P., Learning long-term dependencies by the selective addition of time-delayed connections to recurrent neural network, Neurocomputing, vol. 48, no. 1-4, (2002) pp. 229–250

[19] Mañé, R., On the dimension of the compact invariant sets of certain non-linear maps, Dynamical Systems and Turbulence, vol. 898, (1981) pp. 230-242.

[20] Takens, F., On the numerical determination of the dimension of an attractor, Dynamical Systems and Bifurcations, vol. 1125 (1985), pp. 99-106.

[21] Duro, R. J., Reyes, J.S, Discrete-time backpropagation for training synaptic delay-based artificial neural networks, IEEE Transactions on Neural Networks, vol. 10, no. 4, (1999) pp. 779-789

[22] Michalewicz, Zbigniew. Genetic algorithms+ data structures= evolution programs. Springer, (1996).

[23] Bellas, F., Duro, R.J., Faina, A., Souto, D.: Multilevel Darwinist Brain (MDB): Artifcial Evolution in a Cognitive Architecture for Real Robots, IEEE Trans. On Autonomous Mental Development, vol. 2, no. 4, 340-354, (2010).