Introducing Long Term Memory in an ANN based Multilevel Darwinist Brain

F. Bellas and R. J. Duro
Grupo de Sistemas Autónomos
Universidade da Coruña

Abstract. This paper deals with the introduction of long term memory in a Multilevel Darwinist Brain (MDB) structure based on Artificial Neural Networks and its implications on the capability of adapting to new environments and recognizing previously explored ones by autonomous robots. The introduction of long term memory greatly enhances the ability of the organisms that implement the MDB to deal with changing environments and at the same time recover from failures and changes in configurations. The paper describes the mechanism, introduces the long term mermoy within it and provides some examples of its operation both in theoretical problems and on a real robot whose perceptual and actuation mechanisms are changed periodically.

1 Introduction

The Multilevel Darwinist Brain (MDB) is a proposal for a Darwinist cognitive mechanism intended for autonomous artificial organisms that must generate original solutions and use previously acquired experience when negotiating new environments and situations. The initial ideas for MDB were introduced in [1] and [2], and in this paper we are going to study its capabilities in order to adapt to changing environments.

This cognitive mechanism is based on a series of theories within the field of cognitive science that relate the brain and its operation through a Darwinist process. These theories are: the Theory of Evolutionary Learning Circuits (TELC) [3], [4]; the Theory of Selective Stabilization of Synapses (TSSS) [5], [6]; the Theory of Selective Stabilization of Pre-Representations (TSSP) [7] and the Theory of Neuronal Group Selection (TNGS) or "Neural Darwinism" [8]. Each theory has its own features but they all lead to the same concept of cognitive structure based on the fact that the brain adapts its neural connections in real time through evolutionary selectionist processes.

Taking inspiration from these theories we have developed an operational cognitive mechanism for artificial organisms which does not predetermine their behavior or learning and which allows them to find creative and original solutions in an autonomous way. This mechanism was designed to work in real dynamic environments and, in this kind of problems, it is very important to manage previously learned information efficiently. We have included a Long Term Memory (LTM) block in the basic schema of the MDB, and the study we have carried out on this LTM can be generalized to any kind of learning structure.

Some other authors have made reference to structures that make use of on-line evolutionary techniques, for example [9] and [10], applied to autonomous robotics but in these mechanisms no long term memory was considered.

The paper is structured as follows: in section 2 we will review the main features of the MDB, in section 3 we are going to include a Long Term Memory in the basic schema of the mechanism, in section 4 we will present the results obtained with this LTM and finally some conclusions of the work will be extracted in section 5

2 Multilevel Darwinist Brain

A Multilevel Darwinist Brain (MDB) [1][2], is an approach to implement a mechanism that is able to allow an organism to interact with the environment and learn from it so that it can improve its performance in time without the designer predetermining future behaviours, in a computationally efficient way. To this end we have considered several concepts:

- *Strategies*: sequences of actions applied to the effectors of the agent
- *World model (𝑊)*: function that relates the sensory inputs of the agent in the instant of time t to the sensory inputs in the instant t+1 after applying a strategy
- *Internal model (𝐼)*: is a function that relates the sensory inputs in instant of time t+1 with the internal state according to the motivation for the agent
- A*ction-perception pair*: is a set of values made up by the sensorial inputs and the internal state obtained after the execution of a strategy in the real world. It is used when perfecting world and internal models.

We have connected these elements as shown in the block diagram of figure 1, which represents the whole mechanism. The final objective of the mechanism is to obtain the strategy the agent must execute in the real world to satisfy its motivations. This selected strategy is represented in the block diagram of figure 1 by the block labeled *Current Strategy* and is applied to the *Environment* through the *Effectors* providing new *Sensing* values for the agent. Thus, after each iteration, we have an action-perception pair obtained from the real world. These action-perception pairs are stored in the *Short-Term Memory* and are used as the fitness function for the evolution of the world and internal models.

Three concurrent evolutionary processes take place in our mechanism (figure 1):

- *World Model Evolver:* structure that manages world models through evolution. This structure evolves a population of world models and each moment of time selects the best one according to the information it has about the real world.
- *Internal Model Evolver:* manages an internal model base, evolving it and providing the best internal model available when it is requested.
- *Strategy Evolver:* manages and evolves strategies and when the agent needs one it provides it.

Each one of these evolutionary processes starts from the population stored in a memory (*World Model Memory, Internal Model Memory and Strategy Memory*). The contents of these memories are random for the first iteration of the mechanism. These contents are maintained from one iteration to the next in order to obtain a generally good population of models and not a superindividual. The individual with the highest fitness value after evolution according to the information of the short-term memory is selected as the current model, both in the case of world and internal models. These

current models are represented in figure 1 by the *Current World Model* and *Current Internal Model* blocks.

In the case of strategies, these are tested during their evolution by implementing a virtual environment using the current models, that is, the current world model provides the sensory inputs in instant t+1 for the strategy we want to test. The current internal model uses these predicted values as input to predict the internal state corresponding to the strategy according to the agent's motivation. The strategy with the highest fitness value is selected as the *Current Strategy* and is applied to the *Environment* through the *Effectors*, obtaining again new *Sensing* values and thus a new *action-perception pair* that is stored in the *Short-Term Memory*.

This basic cycle is repeated and, as time progresses, the models become better adapted to the real world and the selected strategies improve in their efficiency to fulfil the agent's motivations.

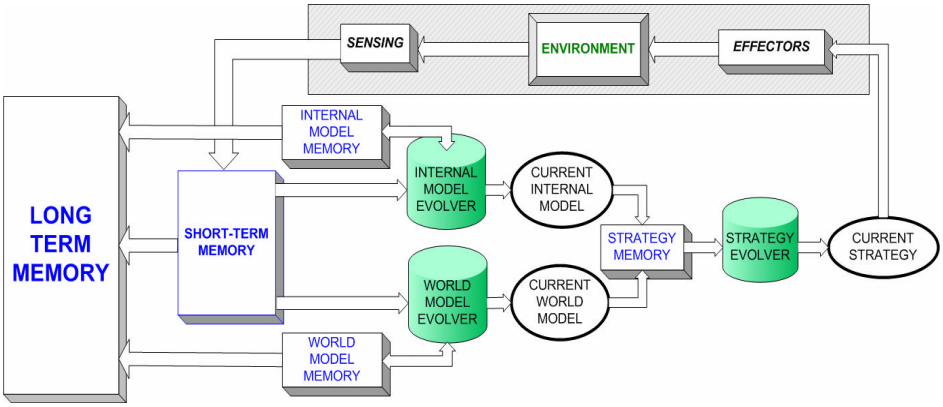

Figure 1. Block diagram of the enhanced MDB

Two elements that must be considered when applying this mechanism are the STM replacement strategy and the evolvers that must be used. Regarding the first of these two points, it must be considered that the information obtained from the real world which is stored in the STM can't, obviously, be infinite. So we need an efficient replacement strategy to manage the stored information.

The main objective must be to store the most relevant data that the environment provides, so the models (world and internal) can predict this environment in the internal operation of the mechanism. The information the agent receives in any given instant of time and which it must use to evaluate its models is partial and, in many cases, noisy or irrelevant. Any model it extracts from this instantaneous information will, in most cases, be useless for other instants of time. To generalize appropriate models the agent will have to consider and relate information obtained in many different instants of time in an efficient manner.

In more mathematical terms and considering a set of data that define a function, the fitness of the model is given by how well it fits the whole function and not by how well it fits the individual point or the subset of points it is considering in a given instant of time. This is what we have called a Sparse Fitness Function (SFF), the

fitness of the individual is given by its response to several different instantaneous local fitness functions, which together conform the desired global one. A more thorough study of this kind of fitness functions is presented in [11], but to summarize we have developed a replacement algorithm for the STM which contains 4 terms:

1. **Distance:** a min-max algorithm is used to decide if an action-perception pair is relevant. This term distributes the information in the problem space.
2. **Functionality:** the information considered is more relevant if it is wrongly-predicted by the current model.
3. **Initial relevance:** if a pair is wrongly-predicted before it is stored in the STM, it must be important to determine the general shape of the generalized function.
4. **Age:** an age term may be included to force the STM to adapt to changes in time.

The replacement mechanism includes these four terms in order to decide if a new pair must replace an existing one in the STM. The weight of each term is adjusted as a function of our requirements of generalization, antiquity, etc. or through the dynamic interaction of the agent with the world, but this is beyond the scope of the present paper.

In addition to managing the STM, when applying the basic MDB to real tasks [2], we have found that our main problem is learning the world models. The required evolutionary process tries to obtain the best ANN to model the environment, and this is complex in real problems where the environment is not regular at all.

We have carried out a lot of experiments with different parameters of the ANNs and GAs in order to find the best solution to a given problem. Finally, we decided to develop a new GA which must autonomously obtain the size of the ANNs representing the world, which must divide a complex function into simple parts automatically obtaining the appropriate number of simple ANNs that make up the complex one and, finally, which must evolve these different models in the same population. The algorithm was called the Promoter Based Genetic Algorithm and was presented in [12].

3 Long Term Memory

At this point, we can include a new term into the basic Block Diagram of the MDB that is shown in figure 1. This new block represents a new memory that stores the world and internal models that were good when used. In more specific terms, when an agent learns to model its environment or just a part of it with accuracy, we would like to store this model in a higher level memory (as compared to the STM). The resulting MDB block diagram is shown in figure 1 labeled as Long Term Memory.

In real problems if the environment where the agent is changes and later it returns to the previous one, it is not necessary to learn the same function again minimizing thus the learning time. We can apply this reasoning to the internal models, because the motivation of a behavior can change in a period of time but older motivations (previously learned internal models) could appear again. In what follows, we will refer just to world models but the results can be generalized to internal models too.

In the current configuration of the MDB, we have a PBGA obtaining models from the information that is stored in the STM using the presented replacement strategy. As

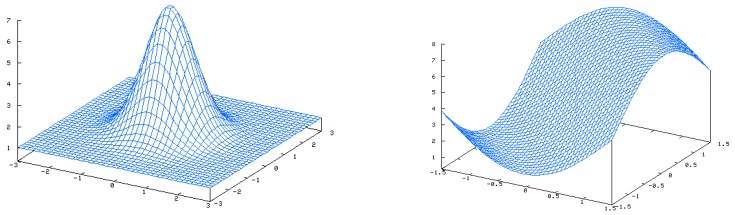

Figure 2. Theorical functions used to test the behaviour of the LTM

we can see, this is a complex mechanism, where once a model is well-learned we don't want to lose it. To decide if a model must go into the LTM, we have to establish when a model is stable in terms of its error in predicting the pairs arriving at the STM. We just want to store in the LTM models that are really general because we would like to have only one model for each different zone of the environment or circumstance. Thus, we have used a stability criterion related to the error provided by the current world model applied to the whole STM in a given instant of time. If this error value is stable (in a 30% window) during a number of interactions equal to the size of the STM, we assume that the model is stable and can be stored in the LTM.

But what happens later? Do we have to look for more stable models? We want to have the best models in LTM, so each time a new model is stable (in terms of variations of prediction error), it must be stored in the LTM. Each time a model is fit to go into the LTM, we compare it with all those already in LTM. To carry out this comparison, we store the model and the STM the model was stable with. Then, we apply the new model to the STM of the existing models and the existing models to the current STM. If both errors are similar, we assume that the models predict the same zone and we leave the best one in LTM. If the errors are very different with the crossed STMs, we assume that we are dealing with a new model and it is automatically stored in the LTM.

All the models in the LTM in a given instant of time are introduced in the evolving population as seeds, so if the agent returns to a previous learned zone, the model will be present in the population and the prediction will be very good soon. Additionally, if the new area is similar to one the robot has seen before, the fact of seeding the population with the LTM models will allow the evolutionary process to reach a solution very fast.

Another element that must be considered is how to detect zone or circumstance changes so that data from two areas are not mixed in the STM thus forcing the

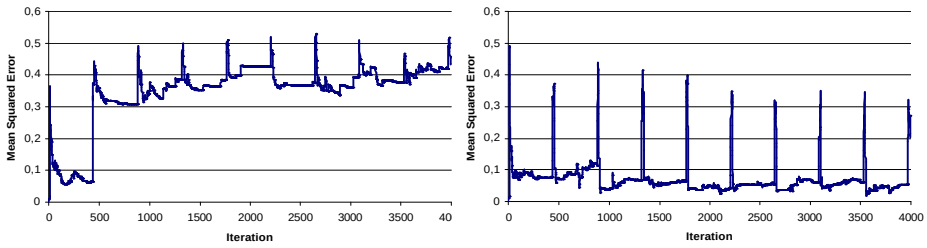

Figure 3. Evolution of the MSE of the prediction made by the current world model. The left figure was obtained witouth LTM and the right figure using LTM.

existence of intermediate models, which is something undesirable. To do this we have just considered an instability criterion for the predictions. When the predictions suddenly become highly unstable for a given period of time, we assume we are in a new zone and purge the contents of the STM by increasing the relevance of the age factor.

Summarizing, we have developed a mechanism to manage the LTM of an evolutionary process that is evolving ANNs to model the world. This mechanism is based on the detection of stable models that are stored in the LTM after applying them to the STMs of the previously stored models (crossed prediction errors). This way we just have one model (the best) for each stable zone of the environment.

4 Experimental results

To test the relevance of the LTM in our evolutionary processes, we first run the MDB using as environment a theoretical 3D function (Gaussian function). Thus we can control the predictions made by the ANNs obtained. After a period of time with the Gaussian, we make the "world" change and it becomes a 3D sinusoidal function. These two functions are displayed in figure 2.

In figure 3 we show the evolution of the mean squared error (MSE) of the prediction made by the best world model without LTM (left) and with LTM (right). As mentioned before, the evolution starts using the gaussian function and, after 450 iterations of evolution, we change it and start using the sinusoidal one. This change of functions occurs every 450 iterations from this point onwards. As we can see in figure 3, the evolution without LTM seems to adapt well to the first change of function, but in the subsequent ones the error increases and the results of modelling are poor. This is because the STM is not large enough to model both functions simultaneously, and the models obtained from the combination of pairs coming from both zones induce an intermediate model that is not good in either zone.

In the right figure, we show the result obtained using the LTM and the results are very good. The evolution of the error is good in the stable zones, there are clear transistions between zones (large error before instability is detected) and this beahaviour is mantained in time. For example, in the first 450 iterations (gaussian

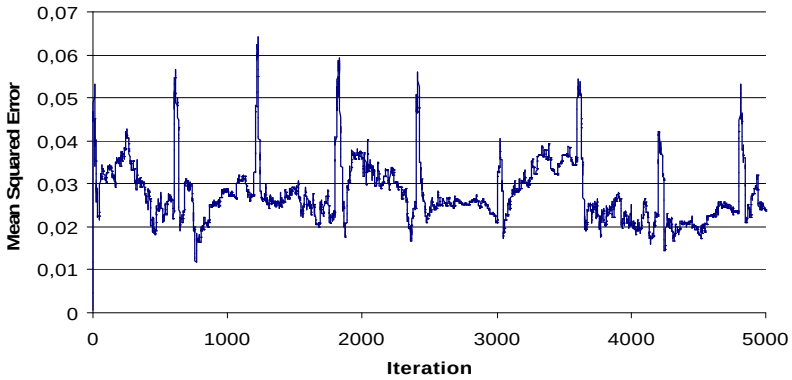

Figure 4. Evolution of the mean squared error of the prediction made by the actual world model in the distance predictions using the LTM in a

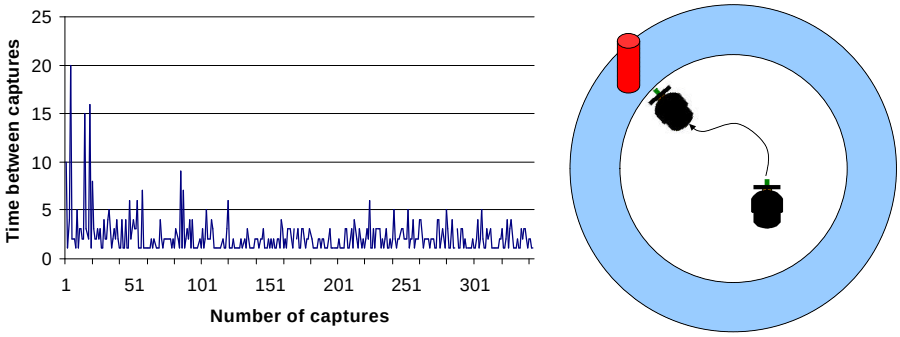

Figure 5. The left graph shows the time between two consecutive captures of the object. The right schema shows the experimental setup

function) the management mechanism stores one model in the LTM and in iteration 900, when we use the gaussian function again; this model is used as a seed in the population. The transition when entering in the gaussian zone the next times is very short because the models were learned before. This way, we reduce learning time maximizing the accuracy of the modeling. The instability detection works as expected and unlike in the case corresponding to the right figure, there is not any mixing of pairs in the STM and thus independent models are obtained automatically for each zone. In addition, it can be observed that as the MDB sees more iterations of the two zones, the models become better, the error is lower, this is an indication that the models corresponding to each zone in the LTM are being changed for better ones when these are available.

The discussion we have presented is focused on applying the MDB to real problems (real robots and real environments). Thus, we think it is necessary to test the whole mechanism in a real task. To do this, we have used the Pioneer 2 robot (a wheleed robot) and considering a simple task of finding an object with its sonar sensors and reaching it, as shown in the right part of figure 5, where we display a real trajectory followed by the robot to reach the objective.

It is a simple task, but the world model must deal with 16 noisy sonar sensors and two noisy actuators (wheels). The world models are represented by ANNs obtained with the PBGA and they have 5 inputs and 3 outputs. The inputs are: distance given by the nearest sonar, angular position of that sonar, a boolean value that permits distinguishing if the sonar is in the front or back of the robot and the two motor values given as linear and angular speed. The outputs are the predicted distance given by the nearest sonar, the predicted angular position of that sonar and the predicted boolean value.

In figure 4 we display the evolution of the mean squared error in the prediction of the first output (distance to the object) made by the current world model in each iteration. Every 600 iterations we assume that a failure in the real robot occurs skewing its perception of the world. Basically the perceptions of distances is inverted in sign and the actions of the wheels are exchanged. Obviously, initially, after the failure a large error appears in the prediction, but after the system has seen the two perception systems once it is able to recover very rapidly and its mean squared error remains between 0,02 and 0,03. In the left part of figure 5 we display the time between two consecutive captures of the objective, initially it is high, but after a while it consistently reaches it in two or three sensings.

Conclusions

In this work we have added a new element, a long term memory, to the Artificial Neural Network based Multilevel Darwinist Brain. This new element increases the power of the cognitive mechanism allowing the agent to remember ANN based world and internal models that have been useful before and incorporate them or even modifications of them into new behaviors due to the fact that the contents of the long term memory participate as seeds in the evolutionary processes within the MDB. The results confirm the usefulness of this strategy within the mechanism in terms of enhancing the capacity of the artificial organism to adapt to new situations and environments it finds itself in as well as to recover from failures and changes in its perceptual and/or actuation capacities.

Acknowledgements

This work was funded by Xunta de Galicia under project PGIDIT02PXIB10501PR the MCYT of Spain under projects TIC2000-0739C0404 and REN2000-0204-P4-0.

References

1. F. Bellas, J.A. Becerra, R. J. Duro, Using evolution for thinking and deciding, Proceedings WSES2001, pp 6161-6166.
2. F. Bellas, A. Lamas, R.J. Duro, Multilevel Darwinist Brain and Autonomously Learning to Walk, Proceedings CIRAS2001, pp 392-398.
3. M. Conrad, Evolutionary Learning Circuits, J. Theor. Biol. 46, 1974, pp. 167.
4. M. Conrad, Complementary Molecular Models of Learning and Memory, BioSystems 8, 1976, pp. 119-138.
5. J.P. Changeux, P. Courrege, A Theory of the Epigenesis of Neural Networks by Selective Stabilization of Sinapsis, Proc. Natl. Acad. Sci. USA, 70, 1973, pp. 2974-2978.
6. J.P. Changeux and A. Danchin, Selective Stabilization of Developing Synapsis as a Mechanism for the Specification of Neural Networks, Nature 264, 1976, pp. 705-712.
7. J.P. Changeux, T. Heidmann., and P. Patte, Learning by Selection. The Biology of Learning, P. Marler and H.S. Terrace (Eds.), Berlin, 1984, pp. 115-133.
8. G.M. Edelman, Neural Darwinism. The Theory of Neural Group Selection. Basic Books 1987.
9. P. Nordin, W. Banzhaf and M. Brameier., Evolution of a World Model for a Miniature Robot Using Genetic Programming, Robotics&Autonomous Systems, Vol. 25, pp. 105-116.
10. L. Steels, Emergent Functionality in Robotic Agents through On-line Evolution, Artificial Life IV, 1995, pp. 8-14 .
11. F. Bellas, J.A. Becerra, R.J. Duro, Sampled Fitness Functions in Complex Problems (parts I and II), Proceedings JCIS 2002, pp 631-639.
12. F. Bellas, R.J. Duro, Modelling the world with statistically neutral PBGAs. Enhancement and real applications, Proceedings ICONIP 2002, pp 2093-2098.