Robustness in the long run: Auto-teaching *vs* Anticipation in Evolutionary Robotics

Nicolas Godzik, Marc Schoenauer, Michèle Sebag

TAO team, INRIA Futurs and LRI, UMR CNRS 8623 bat. 490, Université Paris-Sud
91405 Orsay Cedex, France
{Nicolas.Godzik,Marc.Schoenauer,Michele.Sebag}@lri.fr

Abstract. In Evolutionary Robotics, auto-teaching networks, neural networks that modify their own weights during the life-time of the robot, have been shown to be powerful architectures to develop adaptive controllers. Unfortunately, when run for a longer period of time than that used during evolution, the long-term behavior of such networks can become unpredictable. This paper gives an example of such dangerous behavior, and proposes an alternative solution based on anticipation: as in auto-teaching networks, a secondary network is evolved, but its outputs try to predict the next state of the robot sensors. The weights of the action network are adjusted using some back-propagation procedure based on the errors made by the anticipatory network. First results – in simulated environments – show a tremendous increase in robustness of the long-term behavior of the controller.

1 Introduction

One key challenge of Evolutionary Robotics (ER) [7] is robustness, defined as the ability of the controller to efficiently deal with changing environments and previously unseen situations − in other words, to adapt itself to some real world. One prominent approach aimed at robust controllers in ER is based on the so-called auto-teaching networks[9]. In this approach, the controller is made of two parts, simultaneously optimized by evolutionary algorithms. The first part, referred to as Agent Model, is fixed offline. The second part, the Agent, actually controls the robot; in the same time, the Agent is modified on-line to get closer to the agent model (section 2). This way, evolution constructs a dynamic decision system, the trajectory of which is defined from an attraction center (the model) and a starting point (the agent at time 0).
At this point, two time scales must be distinguished. During the training period, the agent is adjusted to the model, the fitness associated to the pair (agent, model) is computed and will serve to find optimal couples of (agent, model). During the robot life-time, referred to as generalization period, the agent is still adjusted to the model in each time step.
However, for feasibility reasons, the training period only represents a fraction

of the robot lifetime. Therefore, the long term dynamics of the controller is not examined during the training period. This would make it possible for (opportunistic) evolutionary computation to select controllers with *any* dynamics in the long run, compatible with a good behavior in the short run...
This paper first focuses on the long-term behavior of auto-teaching networks, making every effort to reproduce as closely as possible the experimental setting described in [9]. Though results could not be exactly reproduced, interesting phenomena appear. Intensive experiments show that, not infrequently, auto-teaching networks with good fitness (good behavior during the training period) are found to diverge (repeatedly hitting the walls) as time goes by.
This paper proposes another source of adaptation, more stable in the long term than a fixed model, inspired from the cognitive sensori-motor framework [10]. The adaptation of the controller architecture is centered on an anticipation module; the anticipation module predicts the next state of the environment (the sensor values) depending on its current state *and the agent action.* When the actual state of the environment becomes available, the anticipation module provides an error signal that can be used either to modify the controller weights or as an additional input. The important issue is that the world model can here be evaluated with respect to the world itself, available for free (at next time step) during both the training and the generalization period. In opposition, the agent model in the auto-teaching architecture could not be confronted to the "true actions" during the generalization period. Implementations of this architecture, termed *AAA*, for *Action, Anticipation, Adaptation*, demonstrate an outstanding robustness in the long run of the simulated experiments, comparatively to the reference results.

The paper is organized the following way. Section 2 briefly describes the auto-teaching architecture, the goal and experimental setting [9], and presents and discusses the results obtained along the same settings when observing the behavior of auto-teaching networks in the long run. In section 3, the *AAA* architecture is presented, focusing on the anticipation module and its interaction with the decision module. Section 4 reports on the experimental validation comparatively to the reference results. The paper ends with a discussion of these first results, and points out the numerous avenues for research opened by this study.

2 Long Term Behavior of Auto-teaching networks

2.1 Settings

With our apologies for the brevity of this reminder (due to space limitation), this section will recall settings and results obtained in [9] (see also [7]). The architecture of auto-teaching networks involves two modules with identical inputs and topologies, feed-forward neural nets without any hidden layer. During the lifetime of the robot, the first module is fixed, while the second module uses the first module as a predictor and adapts its weights using back-propagation.
During evolution, the training period is made of 10 epochs. At the beginning of

each epoch, both the target and the Khepera robot are set to random positions; the robot explores the arena (60 × 20 cm) during at most 500 times steps, until either it hits the 2cm radius target, increasing its fitness by $500 - t_{\text{hit}}$, or it hits a wall, getting no fitness for that epoch (Note that the target is not "visible"). The main result obtained in [9] was that auto-teaching networks did exhibit an adaptive behavior when the color of the walls changed from black to white.

Building on those results, our plan was to examine how the auto-teaching networks adapt to rapidly changing environments. To this aim, the color of the walls was changed every generation, alternating dark and white walls.
As in [9], we used an ES-like Evolutionary Algorithm in which 20 parents generate 100 offspring. The weights were real-coded genotypes, and their values were limited to $[-10, 10]$. The mutation rate was 10%. However, in order to approach the performances, some slight modifications were necessary: we used a (20+100)-ES rather than a (20,100)-ES; We used Gaussian mutation with fixed standard deviation .5 instead of uniform mutation with amplitude 1; And we used some crossover at rate 40%.

2.2 Training period

A first remark is that we failed to fully reproduce the results in [9], due to the very high variability of the fitness with respect to the starting positions of the robot and the target. This variability was clearly visible when post-processing the best individuals from the last generation: out of 10 epochs, it never came even close to the same fitness than it had been given during its last evaluation. The only way to get over that variability and to come close to that was to run 10 times 10 epochs and to take the best results out of those 10 evaluations.

Nevertheless, the on-line results (best and average of 11 runs) resemble those of [9] with higher variance, as can be seen on Figure 4-left, and show a rapid increase toward a somehow stationary value of 2500. However, when we started to investigate the behavior of the controllers over a large number of epochs, we discovered drastic changes after the 10 epochs of "normal" lifetime – most robots starting to repeatedly hit the walls (Figure 4-right).

2.3 Generalization period

Figure 1-right and Figure 2 show typical results in our setting for that experiment. For each epoch (x coordinate), a bar shows whether the robot found the target (positive bar, the smaller the better) or hit the wall (negative bar). To make the figure more readable, no bar is displayed when the robot neither found the target nor hit the wall. The "most decent" individual (Fig. 1-left) only starts hitting the walls after 300 epochs − though it does not find the target very often after the initial 10 epochs. Figure 2 is a disastrous individual, that starts crashing exactly after 10 epochs. More precisely, the best individuals (gathered after a re-evaluation on 10 times 10 epochs of all individuals in the final population) hit the walls on average for 400 epochs out of 1000, the one displayed on Figure 1-left being the best one with only 142 crashes.

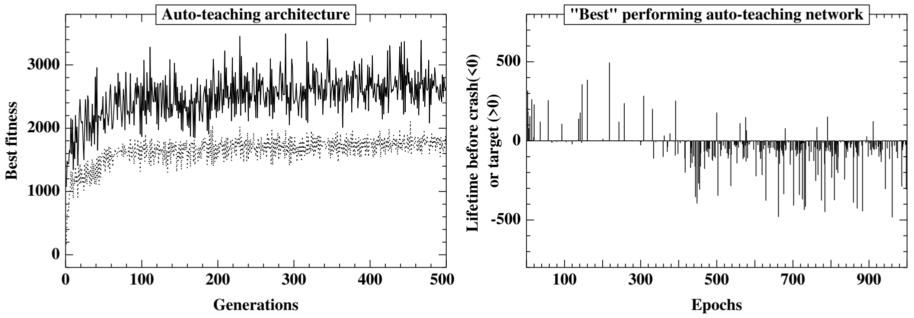

Fig. 1. Experiments with auto-teaching architecture: Left - On-line results (peak and average from 11 independent runs). Right - Life-times for the “Less Disastrous” results on long-term adaptation: negative values indicates that the epoch ended with a crash.

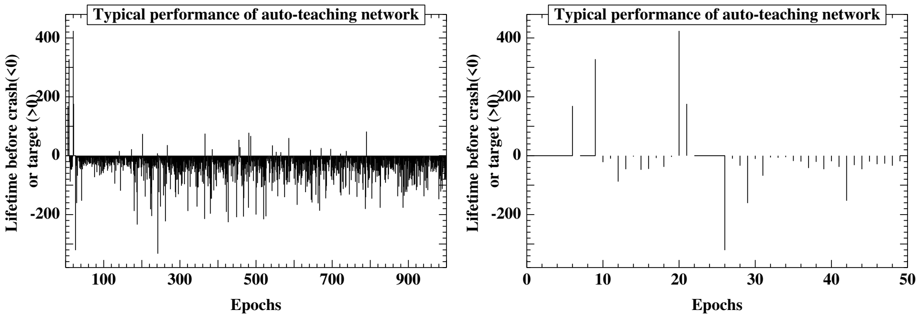

Fig. 2. More typical behaviors on long-term runs: Left - for 1000 epochs. Right - Zoom on the first 50 epochs.

The interpretation offered for these results is that once again, evolution found a mouse hole to reach the goal: because what happens after the training period does not influence selection, anything can indeed happen then. The underlying dynamical system modifies the weights according to some evolved model that only has to be accurate during 10 epochs.

A totally unrealistic solution would be to increase the training period by an order of magnitude during evolution. The anticipatory architecture presented in the next section is an attempt to address the above limitations.

3 Action, Anticipation, Adaptation

The AAA architecture for neural controllers achieves three tasks: action (controlling the robot effectors); anticipation (based on the robot sensors, and the action output); adaptation (based on the difference between the sensor values anticipated in the previous time step, and the current sensor values).

As mentioned in the introduction, the basic underlying idea of the AAA architecture is that the adaptation mechanism must apply only when needed, and must be based on “true errors” rather than errors coming from an arbitrary model purposely evolved. In other words, rather than build a declarative model of the world, the idea is to give the robot a procedural model that will allow him to predict the consequence sof his own actions. And the simplest description of these consequences is through the values of its sensors. Such views are inspired from the cognitive sensori-motor framework [10].

In the framework or neural controllers, and in the line of auto-teaching networks [9], a second neural network, the Model network, is added to the Agent controller, and its goal is to predict the values of the robot sensors at next time step. The inputs of this Model network are both the hidden layer of the actuator controller and the actual commands given to the actuators. Its outputs are, as announced, the values of the sensors (see the dark gray part of Figure 3).

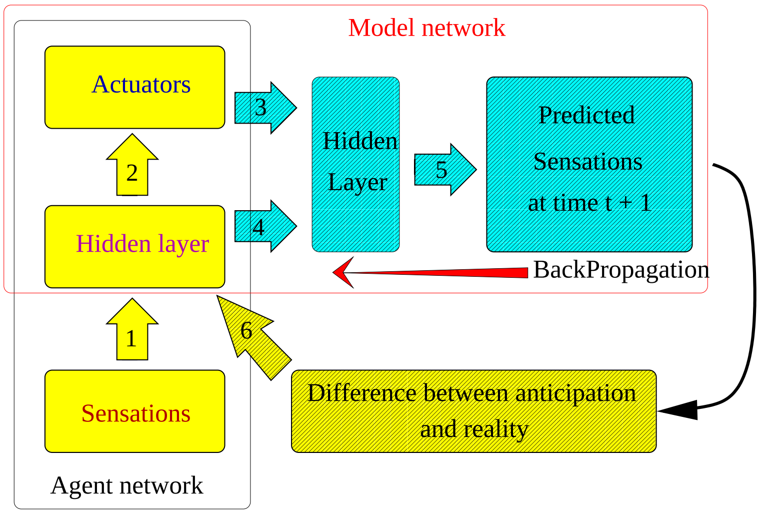

Fig. 3. The complete Anticipatory Neural Architecture. Rounded boxes are neurons, large arrows are connection weights. The classical Agent network gives actuator commands from the sensations. The Model network predicts values for the sensors from the intermediate layer of the actuator network and the actual actuator commands. Back-propagation is applied using the prediction error to all weights backward: 5, then 3 and 4, then 2, then 1. The prediction errors are also added as inputs to the Agent network.

Those predictions are then compared with the actual values sent by the sensors at next time step, and the results of these comparisons are used for a back-propagation algorithm that adjusts the weights of both the Model and the Agent networks, as described on Figure 3. Another possible use of those errors, not used in the experiments described in this paper, is to add them as direct inputs to the actuator network (light gray part of Figure 3). The first author's PhD will consider such alternate architecture.

Note that similar architectures had been proposed in [4], in the framework of robot arm controllers, as an alternative to directly building the inverse problem.

4 Long Term Robustness of *AAA* Architecture

4.1 The neural network

In the experimental framework considered in this paper, the Khepera has 4 inputs (pairs of infra-red sensors), 2 outputs (two independent motors). Hence the anticipatory part of the network must also have 4 outputs. Both hidden layers of the actuator network and of the anticipatory network have 5 neurons. Considering that all neurons also have a bias as input, the resulting network hence has 5×5 (arrows 1 and 6 + bias on Figure 3) + 6×2 (arrow 2 + bias) weights on the actuator network, plus 8×5 (arrows 3 and 4 + bias) + 6×4 (arrow 5) weights for the anticipatory network – 101 weights altogether. All those weights are submitted to back-propagation when some error occurs on the sensor predictions.

4.2 Long-term robustness

The same simulated experiment than that of [9] was run with the *AAA* Architecture. The learning curves along evolution are given on Figure 4-left, averaged on 11 independent runs: they are not very different from the same plots for the auto-teaching network (Figure 4-left). But the results about long-term adaptation are by no way comparable. Whereas the auto-teaching networks show unpredictable behaviors after the initial 10 epochs the anticipatory controllers stay rather stable when put in the same never-ending adaptation environment (e.g. the color of the walls change every 10 epochs, while adaptation though back-propagation is still going on). A typical summary of the behavior of the best individual of an evolution of the anticipative architecture can be seen on Figure 4-right: apart from a few crashes due to starting positions very close to the wall, almost no crash occurs in that scenario.

More precisely, out of 11 independent runs, 8 never crash in 1000 epochs (plots not shown!) while the 3 others had a behavior similar to that displayed on Figure 4-right: they never crash when the walls are white, and start hitting the dark walls after 700-800 epochs of continuous learning.

All those results clearly demonstrate that the anticipatory architecture does not suffer from the auto-teaching networks' defects, and exhibit very stable behaviors even after thousands of epochs (the 8 crash-free best individuals were run up to 50000 epochs with no crash at all).

4.3 Adaptation in *AAA* networks

An important issue, however, is that of the adaptivity of the anticipatory architecture. Indeed, more sophisticated architectures than the simple auto-teaching network described in section 2.1 (like for instance a 3 layers network with one fully recurrent hidden layer) can be evolved to be robust in both the white and black environment – the robots will simply stay further away from the walls in

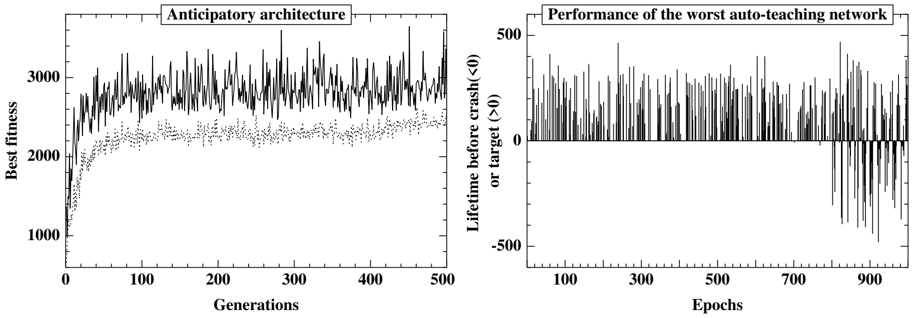

Fig. 4. Experiments with the anticipating architecture: Left - On-line results (average best and average from 11 independent runs). Right - **Worst** result on long-term adaptation (the wall color changes every 10 epochs).

the white environment. But such architectures do not have any adaptive mechanism, and the experiments presented now will demonstrate that the anticipatory architecture does behave adaptively.

The initial a posteriori experiments described in previous section (let the robot live for 1000 epochs, alternating dark and white walls every 10 epochs) did not give any evidence of adaptivity: all anticipatory controllers behave similarly, crashing very rarely against the walls, and behaving almost the same in both dark and white environments: due to the large number of weights, their values change rather slowly.

It was hence decided to let the weights adjust during 100 epochs in the same environment, and some interesting phenomena started to appear. First, after 100 epochs, some individuals began to have trajectories like the ones plotted on Figure 6: whereas the initial weights allow a cautious behavior in case of dark walls (the robot stays farther from the walls, see the thick line on the top plot), this is no longer the case after 100 epochs of weight modification, as witnessed by the bottom plot of Figure 6, where the red cross indicates that the robot hit the wall (dark walls) while it still avoids the walls when they are white. Indeed, after 100 epochs of adaptation to the white walls, the immediate epochs in the dark environment always resulted in a crash. Note that the reverse is not true, and individuals that have spent their first 100 epochs in the white environment never hit any wall, black or white afterward.

But more importantly, after some time in the dark environment, the behavior of the robot comes back to collision-free trajectories. Figure 5 shows two situations in which this happens. When, after the initial 100 epochs in the white environment, the walls remain black forever (left), the number of crashes gradually decreases, and no crash takes place during the last 100 epochs. More surprisingly, when the wall change color every 100 epochs, the rate of crashes also decreases (Figure 5-right), and in fact, it decreases even more rapidly than in the previous scenario – something that requires further investigations.

Note that control experiments with the auto-teaching networks still exhibited an enormous amount of crashes, whatever the scenario.

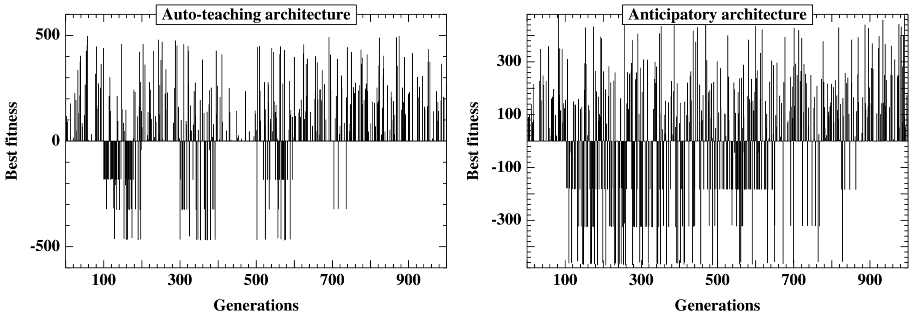

Fig. 5. Adaptation by the Anticipating architecture: 100 epochs are run with white walls; the robot is then put in the dark environment for 900 epochs (left) or is put alternatively in dark and white environments by periods of 100 epochs (right).

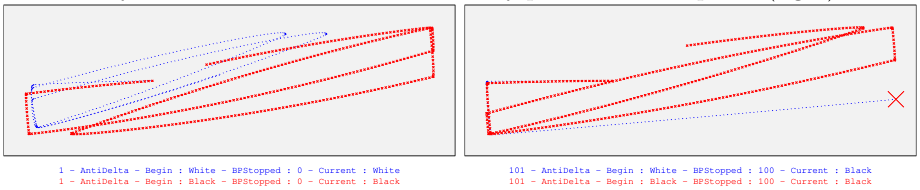

Fig. 6. Trajectories of the best individual of an evolution of an anticipative architecture, where the thick lines correspond to dark walls and the thin line to white walls. The stating points are the same for all trajectories. Left: Initial behavior during the first epoch, before any adaptation could take place. Right: behavior after 100 epochs of adaptation in the white environment.

5 Discussion and further work

The idea of using anticipation to better adapt to changing environments is not new, and has been proposed in many different areas. Anticipatory Classifiers Systems [11] are based on anticipation, but in a discrete framework that hardly scales up.

Trying to predict some other entity's action also amounts to anticipation, but does not really try to anticipate on the consequences of one's own actions (e.g. a program playing "psychological" games [1], or multi-agent systems [13]). Trying to directly predict its own sensor values has also been tried to help building Cognitive Maps in Robotics [2]: the prediction error is then used as a measure of interest for an event.

But the architecture the most similar to *AAA* has been proposed in the Evolutionary Robotic domain by Nolfi, Elman and Parisi [6] in a simpler framework, for artificial organisms foraging food. First of all, no long term experiment was described in that work. Moreover, their architecture did not use the prediction errors as supplementary inputs - but it did use the last output commands . . . Furthermore, the sensors and the actuators were closely related: the sensory inputs were the direction and distance of the nearest food, while the commands for the actuators were . . . the angle and distance to advance: in our experiments, the only relationship between the actuator commands and the predicted outputs is through the Agent network itself.

Another interesting issue is that of the outputs of the Model network. It has been argued by Nolfi and Parisi [8] that the best teaching inputs are not the correct answers for the network (i.e. the exact predictions of the next sensor values). But this might be because of that link mentioned above between the predicted outputs and the actuator commands. Indeed, some preliminary investigations inside the *AAA* neural networks during the lifetime of the robot seem to show that its predictions are here rather accurate most of the time: for instance, when the robot is far from any obstacle, the predicted values are indeed very close to 0 (and hence not modification of the weight does take place). But here again deeper investigations are required.

Looking at the behavior of adaptive systems from a long-term perspective asks new questions beyond the traditional debate between Nolfi's model of interaction between learning and evolution [5] and Harvey's claim that the success of learning + evolution only comes from the relearning of weights that have been perturbed by some mutation [3]. Indeed, the successful re-adaptation observed after a long period in the white environment (Section 4.3) seems to suggest that the learning is not limited to correcting some weight modifications. However, more work is needed to understand how such re-adaptation has been made possible by evolution. In particular, a detailed monitoring of how the weights adapt on-line should bring arguments to this debate.

Another important issue is that of the scaling up of the *AAA* architecture with the number of sensors (e.g. if the robot is equipped with some vision system). A possible answer might come from the *information bottleneck* theory [12]: this model tries to compress the sensor information as much as possible, while still maintaining feasible a reconstruction of the world that is sufficient for the task at hand. In that perspective, the hidden layer of the Agent network (Figure 3) could then be viewed as the set of *perceptions* of the robot, and the Model network could then try to predict this minimal compressed information rather than the numerous sensor values.

Finally, running the *AAA* architecture in real-world environment will be the ultimate validation of the approach. However, the experiment used in this paper is not likely to be easily portable to real-word, because of the huge variance of the results. Using the amount of area swept by the robot seems a better idea, and preliminary results (in simulation) suggest that the phenomena also happen with such more stable fitness.

6 Conclusion

After having pointed out a major weakness of auto-teaching networks, the unpredictability of their long-term behavior, we have proposed the *AAA* architecture to remedy this problem: the evolved oracle of auto-teaching networks without any grasp on reality, is replaced by a Model network that will learn to predict the values of the sensors of the robot. The modification of the weights of the

Agent network (the motor control) is then based on the errors made for those predictions. The first results in terms of long-term robustness are outstanding compared to those of the auto-teaching networks. Moreover, at least some of those networks do exhibit a very interesting adaptive behavior: after having evolved during 100 epochs in a white environment, they can gradually re-adapt to dark walls.

However, a lot of work remains to be done to assess the efficiency and usefulness of the *AAA* architecture, starting with a better understanding of how and why such anticipatory networks can re-adapt their weights on-line without any direct incentive or reward for collision avoidance. Forthcoming experiment will involve other variants of the *AAA* architecture (e.g. adding the error on the prediction as inputs to the controller), more meaningful scenarios (e.g. tracking regularly moving objects) and more importantly precise monitoring of the weight adaptation in different situations for some evolved controllers. We nevertheless hope that anticipatory networks can somehow help bridging the gap between fully reactive controllers and sensori-motor systems.

References

1. Meyer C., Akoulchina I., and Ganascia J.-G. Learning Strategies in Games by Anticipation. In *Proc. IJCAI'97*. Morgan Kaufmann, 1997.
2. Y. Endo and R.C. Arkin. Anticipatory Robot Navigation by Simultaneously Localizing and Building a Cognitive Map. In *ICOS'03 – Intl. Conf. on Intelligent Robots and Systems*. IEEE/RSJ, 2003.
3. I. Harvey. Is there another new factor in evolution? *Evolutionary Computation*, 4(3):311–327, 1997. Special Issue: 100 Years of the Baldwin Effect.
4. M. Jordan and D. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16, 1992.
5. S. Nolfi. How learning and evolution interact: The case of a learning task which differs from the evolutionary task. *Adaptive Behavior*, 7(2):231–236, 2000.
6. S. Nolfi, J.L. Elman, and D. Parisi. Learning and evolution in neural networks. *Adaptive Behavior*, 3(1):5–28, 1994.
7. S. Nolfi and D. Floreano. How co-evolution can enhance the adaptive power of artificial evolution: implications for evolutionary robotics. In P. Husbands and J.A. Meyer, editors, *Proceedings of EvoRobot98*, pages 22–38. Springer Verlag, 1998.
8. S. Nolfi and D. Parisi. Desired answers do not correspond to good teaching input in ecological neural networks. *Neural Processing Letters*, 2(1):1–4, 1994.
9. S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 5(1):75–98, 1997.
10. J. Kevin O'Regan and Alva Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences*, 24(5), 2001.
11. W. Stolzmann. An introduction to anticipatory classifier systems. In P.L. Lanzi et al., editors, *LCS'99*, pages 175–194. LNAI 1813, Springer Verlag, 2000.
12. N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. 37-th Allerton Conf. on Communication, Control and Computing*, pages 368–377, 1999.
13. M. Veloso, P. Stone, and M. Bowling. Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer. In *SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, 1999.