# LEARNING TO CONTROL AN AUTONOMOUS ROBOT BY DISTRIBUTED GENETIC ALGORITHMS

**Marco Colombetti**
Progetto di Intelligenza Artificiale e Robotica
Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci, 32
20133 Milano, Italy
colombet@ipmel2.elet.polimi.it

**Marco Dorigo**
International Computer Science Institute
1947 Center Street
Suite 600
Berkeley, California 94704-1105
USA
dorigo@icsi.berkeley.edu

## Abstract

Machine learning shows promise as a method for developing autonomous robots. In this paper, we report the results of a few experiments carried out both in simulated environments and with a real moving robot. A parallel implementation of a learning classifier system, ALECSYS, is used to shape an agent to perform Animat-type tasks, like chasing a prey or escaping a predator. In particular, we concentrate on the role of innate architecture for scaling up the size of learnable tasks. We show how a relatively complex behavior pattern can be learned as the composition of basic behaviors. A coordination behavior controls the interactions between the basic behaviors.

## 1. Introduction

Machine learning and autonomous robotics are becoming more and more tightly coupled. People from the machine learning community find the autonomous robot problem – or its version known as the Animat problem (Wilson, 1987) – a challenging issue to test new algorithms and methodologies. On the other hand, many researchers in "traditional AI" robot planning believe that learning is going to be necessary to program robots, and also scientists like Brooks (1991b, 1992), who proposed an engineering approach to robot building, recognize the importance of learning to give robots adaptive capabilities.

Our long term research goal is the design of autonomous systems with learning capacities. Even if many different machine learning schemes are known today, learning complex robot control programs is still a very difficult task. In Dorigo & Schnepf (1993), we proposed the use of a straightforward engineering approach to reduce the complexity of learning tasks: we designed modules dedicated to the solution of very simple learning problems; each module had access to a limited amount of sensorial information and learned to propose actions according to its local goals. Such organization can be viewed as a possible implementation of a theoretical model of behavior organization like the one proposed by Tinbergen (1966); or, it can be regarded as a mere engineering approach aiming at the construction of complex learning systems by an appropriate composition of simpler modules, as proposed in Colombetti & Dorigo (1992).

In order to have a flexible tool to build such learning systems, we designed and implemented ALECSYS, a distributed learning classifier system (for a technical description see Dorigo, 1992c). ALECSYS is designed to permit machine learning researchers to carry out experiments in rather complex domains, building hierarchies of classifier systems (CSs). In our approach, each of the learning modules is a CS working on a rule base acquired by interaction with the environment. The learning algorithms, based on genetic rule-discovery, allow the rule base to improve incrementally, guided by feedback received as a result of interaction with the environment.

In fact, we are using ALECSYS to devise new ways of using the learning classifier system, proposed by Holland & Reitman (1978), as a framework for "shaping" an agent to survive in a given environment. Some experiments using ALECSYS to build a simple hierarchical learning system were presented in Dorigo & Sirtori (1991) and in Dorigo & Schnepf (1993). Further results in that direction can be found in Colombetti & Dorigo (1992). In this paper we are interested in the application of ALECSYS to controlling the *AutonoMouse*, an autonomous mouse-shaped robot.

The paper is organized as follows. Section 2 briefly illustrates related work, both in autonomous robotics and in reinforcement learning. In Section 3 we sketch our approach to building adaptive systems. In Section 4 we illustrate the experimental setting in a simulated environment and report results obtained by the learning agent. Section 5 is devoted to the discussion of results obtained by a real robot in a simpler, but real, environment. Finally, in Section 6 we draw some conclusions and indicate the future directions of our research.

## 2. Related work

Recently, much work has been carried out in *reinforcement learning*. Reinforcement learning is a kind of machine learning in which all the feedback information the learning system receives is in the form of positive or negative numbers, respectively called *rewards* and *punishments*. This

means that only a minimal amount of domain knowledge is given to the system[1]. Reinforcement learning is generally *unsupervised*, i.e. new examples are created by the explorative activity of the learning agent, and are not chosen *a priori* by a teacher. Nevertheless, a teacher is sometimes used, either to give rewards, or to help through the presentation of a feasible (or optimal) solution (see for example reinforcement learning with teaching in Lin, 1991, 1992).

Reinforcement learning has been studied in different algorithmic frameworks. Notably, we have connectionist reinforcement learning (e.g., Barto, Sutton & Anderson, 1983; Williams, 1992); classifier systems reinforcement learning (e.g., Holland & Reitman, 1978; Robertson & Riolo, 1988; Booker, 1988; Dorigo, 1992b); temporal differences reinforcement learning and related algorithms, like the adaptive critic heuristics (Sutton, 1984) and Q-learning (Watkins, 1989, 1992). Often, adaptive critic heuristics and Q-learning are implemented by means of a connectionist system (e.g., Lin, 1992).

Most of the applications used to illustrate and compare the proposed algorithms are taken from the realm of autonomous robotics. A few of them deal with real robots, most with simulated ones. In many cases, the application problem is a subset of the Animat problem, that is the issue of realizing an artificial system able to adapt and survive in a natural environment.

In our work we define a number of *basic* and *coordination* behavioral modules, we combine them in a hierarchical architecture, and then we make the system learn basic behavioral patterns and coordination policies. To guide the learning process, we use a *shaping procedure*, which is reminiscent of the procedures used by laboratory psychologists to shape experimental subjects. We use reinforcement to shape basic behaviors first, and then the coordination behavior.

Many researchers have advocated some kind of architecture (see for example Brooks's subsumption, 1991a). Some of them do not rely on learning (Brooks), others use learning only for basic behaviors (Mahadevan & Connell, 1992; Mahadevan, 1992). Still other researchers (see for example Lin's application of Q-learning, Lin, 1992) do not face the problem of task factorization, and learn by some "flat" system – that is, systems with no explicitly built-in architecture. At the authors' knowledge, our project is the only one using learning to coordinate learned behavioral modules, and also the only one exploiting classifier systems to shape real robots.

1 It is impossible to provide no knowledge at all about the problem domain. The learning system designer has at least to define the interface with the environment, which is itself determined by the problem structure. It is also the case that, in systems like ours, a certain amount of information is put into the learning system architecture. This means that we solve the problem of the system architecture, that in nature was solved by evolution.

## 3. Our framework

In our learning system there are basically two learning entities: *basic behaviors* and *coordination behaviors*. They are implemented and composed to build a hierarchical architecture using ALECSYS. ALECSYS, introduced in Dorigo & Sirtori (1991) and in Dorigo (1992c), is a tool that allows one to distribute classifier systems on a net of transputers. Through ALECSYS, a transputer net can host many CSs, and each CS can in turn be distributed on a subnet of the transputer net.

In our work we have been influenced by Wilson's Animat problem. This accounts for our interest in behavioral patterns that are the artificial counterparts of basic natural responses, like feeding and escaping from predators. Our experiments are therefore to be seen as possible solutions to fragments of the Animat problem. Moreover, we are also interested in applying to real robots the results obtained with simulated Animats. Although we believe that experiments should be carried out in the real world to be truly significant, such experiments are in general costly and time-consuming; therefore, we use simulations to preselect a small number of potentially relevant experiments to be performed with the real robot.

One of the hypotheses we explore is that relatively complex behavioral patterns can be built bottom-up from a set of simple responses. In this paper we consider three kinds of basic responses:

- the *approaching behavior*, i.e. approaching an (almost) steady object with given features; in the natural world, this response is a fundamental component of feeding and sexual behavior;
- the *chasing behavior*, i.e. following and trying to catch a moving object with given features; as the preceding approaching behavior, this response is important for feeding and reproduction;
- the *escaping behavior*, i.e. moving as far as possible from an object with given features; the object can be viewed as a predator.

Other kinds of behaviors have been the subject of previous work (Dorigo & Sirtori, 1991; Dorigo, 1992a); in particular:

- the *mimetic behavior*, i.e. entering a well-defined physical state which is a function of a feature of the environment; this is inspired by the natural behavior of a chameleon, changing its color according to the color of the environment;
- the *avoidance behavior*, i.e. avoiding physical contact with an object of a given kind; this can be seen as the artificial counterpart of a behavioral pattern which allows an organisms to avoid hurting objects.

More complex behavioral patterns can be built from these simple responses in many different ways. So far, we have studied the following building mechanisms (composition rules):

- *Independent sum*: two or more independent responses are produced at the same time; for example, an agent may assume a mimetic color while chasing a prey.
- *Combination*: two or more homogeneous responses are combined into a resulting behavior; consider the movement of an agent which is following a prey and trying to avoid an obstacle at the same time.
- *Suppression*: a response suppresses a competing one; for example, the agent may give up chasing a prey in order to escape from a predator (suppression is similar to subsumption, as studied by Brooks, 1991a, and by Mahadevan & Connell, 1992).
- *Sequence*: a behavioral pattern is built as a sequence of simpler responses; for example, fetching an object involves reaching the object, grasping it, and coming back (sequences have been extensively studied by Singh, 1992a,b).

In general, more than one mechanism can be at work. Consider an agent trying to avoid steady hurting objects while chasing a moving prey and being ready to escape if a predator is perceived. In this case, the chasing behavior will be combined with obstacle avoidance; at times, the escaping behavior will suppress the chasing behavior (but still combine with obstacle avoidance!).

## 4. Experiments in the simulated environment

Given the framework of Section 3, a number of different experiments can be carried out. Some experiments regarding the learnability of basic behaviors and of coordination behaviors have been discussed in Dorigo & Sirtori (1991), in Dorigo & Schnepf (1993), and in Colombetti & Dorigo (1992). In this paper, we start to investigate the scalability of our approach, when the whole behavior increases in complexity. Below we report on the following experiment.

Consider an agent who should learn the following behavior:

```
If there is a predator
   then Escape
   else if hungry
           then Feed
           else Chase the moving object
```

In the previously cited papers we showed that the correct behavior can be easily learned by a hierarchical structure in which basic modules learn basic behavioral patterns and coordination modules learn the coordination policy. Reinforcement is given to the learning system by a program that implements the above described behavior (for example, there is a positive reward if the Animat moves in such a way that causes its distance from a predator to increase, etc.). In Figure 1 we report an example of the input-output interface for the chasing behavior[2] (CS-Chase). The input pattern, i.e. a message the learning system receives from its sensors, gives a coarse code of the chased object position. (The Animat has four on/off eyes, one on each side of its square shape. Each eye is set to 1 when the object is seen, to 0 when it is not seen.) The output pattern is made up of two components; the first one is a proposed action (in which direction to move, and whether to move or not; it is sent to the effectors), while the second one is a message sent to the coordinator, to let it know that CS-Chase was proposing an action.

The architecture we used to combine the three behaviors is sketched in Figure 2. This is only one of a set of possible choices which are described in detail in Colombetti & Dorigo (1992). Essentially we have three basic CSs, whose task is to learn the basic behavioral responses. We also have a coordinator CS, whose task is to learn to coordinate basic CSs. Basic modules send to the coordinator a bit string (in this application a single bit) whose meaning is learned. This means that, when the system has learned, each basic module will set the bit sent to the coordinator to a particular value when it wants to do something. For example, CS-Escape could learn to set the bit to 1 whenever it sees a predator. CS-coordinator then learns some kind of composition rule, which gives different weights to the actions proposed by basic CSs. In our example the composition rule learned is *suppression*, meaning that only one out of the three basic behaviors has the right to control the learning system effectors.

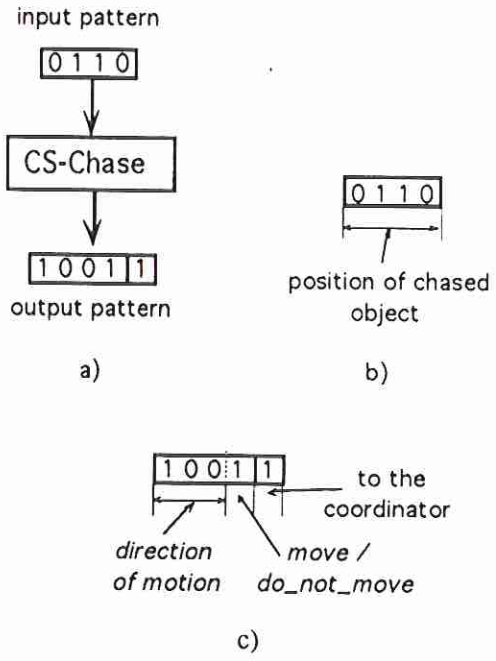


Figure 1. a) Example of input-output interface for the CS-Chase behavior;
b) Example of input message;
c) Example of output message.

[2] CS-Feed and CS-Escape have a very similar structure.

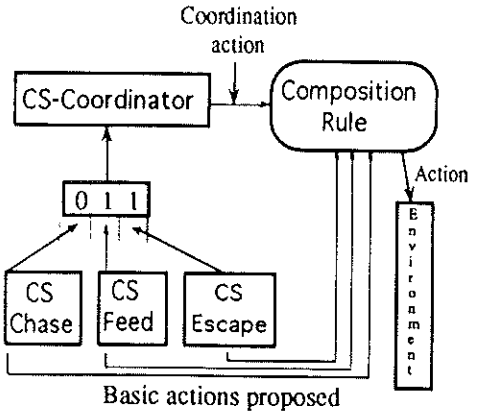


Figure 2. Example of hierarchical architecture for a three behaviors learning task.

We now make the problem more difficult by composing the coordination learning problem with a discrimination learning problem. In Dorigo (1992a), we showed that CSs can be used to discriminate between input patterns. They can be used for example to discriminate between similar objects, to learn which of them are food and which are not. In this new experiment we let the learning agent perceive many different objects for each class; besides learning to do the right thing, the agent must also learn to choose which, among the different objects in a class, is the relevant one. We label each class of objects with names that remind the role they play in our example: *possibly-dangerous-animals*, *food-like-objects* and *possibly-interesting-moving-objects*. The behavior to be learned becomes:

```
If there is a predator in possibly-dangerous-
   animals
   then Escape
   else if there is food in food-like-objects
           then Feed
           else Chase a particular object in
                possibly-interesting-moving-
                objects
```

The two simulated environments in which our Animat lives are sketched in Figure 3. In the next section we present some results obtained with our system in both the coordination task and in the coordination task with discrimination. Experiments were run in both environments. The resulting performance of experiments run in the environments of Figures 3a and 3b are reported respectively in Figures 4a and 4b. Performance is measured as the ratio of the number of correct responses to the total number of responses produced from the beginning of the simulation. Clearly the performance is always $\leq 1$. In both experiments we shaped the robot in three phases. In the first phase, we shaped the basic behaviors; in the second phase, we froze the basic behaviors (i.e., we deactivated the learning algorithms), and started to shape the coordination behavior[3]. Finally, in the last phase we let all the system free to go on learning. Results show that the first phase was more difficult in the case of the environment of Figure 3b. This is clearly due to the discrimination task, which made the search space much larger. Nevertheless, if we give enough time to the system to learn the basic behaviors, the following phases lead to comparable overall performance.

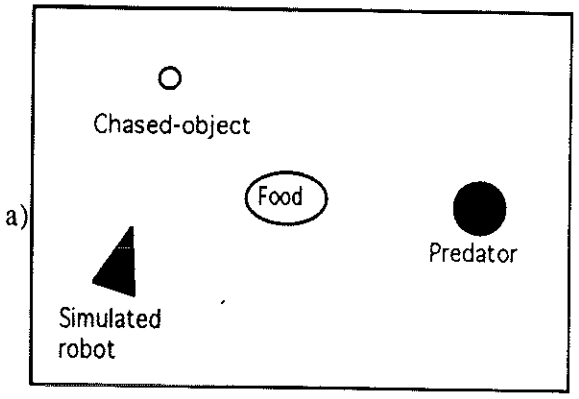


Figure 3. a) Simulation environments: One object in each class (no discrimination is necessary).

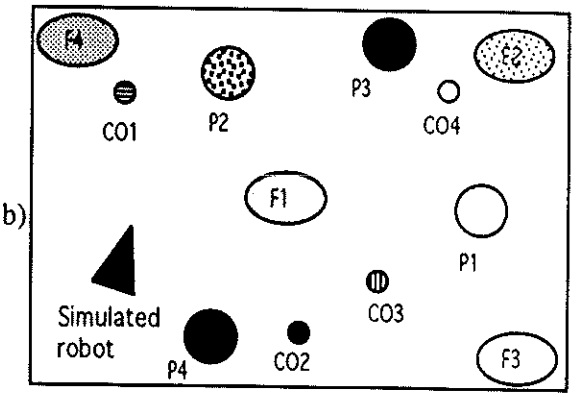


Figure 3. b) Simulation environments: Four objects in each class (discrimination becomes necessary).

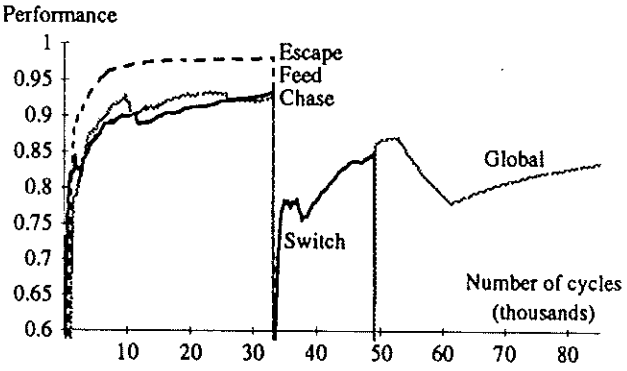


Figure 4. a) Performance of the typical experiment with environment of Figure 3a. Between cycles 0 and 33,000 basic behaviors were shaped; between 33,000 and 49,000 basic behaviors were frozen and the coordinator (switch) was shaped; after cycle 49,000 the system was free to go on learning.

[3] In this experiment the coordination module is also called *switch* because its task is to choose one of the basic behaviors.

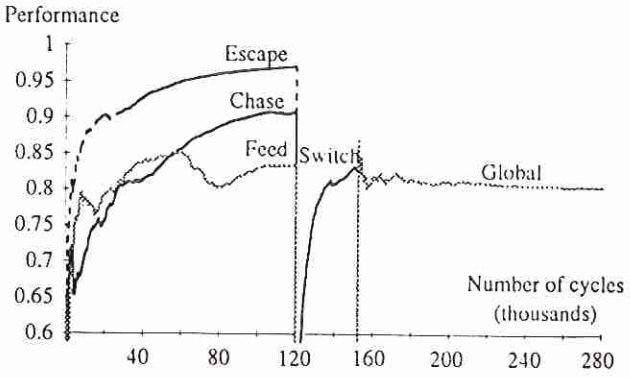


Figure 4. b) Performance of the typical experiment with environment of Figure 3b. Between cycles 0 and 120,000 basic behaviors were shaped; between 120,000 and 152,000 basic behaviors were frozen and the coordinator (switch) was shaped; after cycle 152,000 the system was free to go on learning.

## 5. Some experiments with the real robot

As it is often the case, experiments with the real robot are not so sophisticated as the ones in the simulated environment. This is mainly due to scheduling constraints: we had first to evaluate simulations results, and then to apply the best performing algorithms to the real robots (we call our robot *AutonoMouse*, due to its autonomous nature and mouse size). So, up to now, experiments with the AutonoMouse have been run only for a single behavior environment. The task we chose for our experiments is the *light chasing* task. The AutonoMouse can perceive a light source and should learn to approach (or follow) it. Light is sensed by means of two on/off eyes which are positioned on the front of the AutonoMouse (see Figure 5). Each eye senses the light within a cone of about 60 degrees. There can therefore be situations in which the light is on, but the AutonoMouse must turn to see it. In this and in the following experiments the performance was evaluated through light intensity, detected by a central light sensor positioned on the robot. This sensor could discriminate 256 levels of light intensity. Inherent uncertainty associated with sensors is not modeled.

In the graphs of Figures 6 and 7 we also plot the average reward over intervals of 20 cycles, multiplied by 25 to ease visual comparison. Figure 6 shows the result of the light approaching experiment. The drop in performance at cycle 225 is due to a sudden change in light position caused by the experimenter moving the light far away from the robot. The big difference in the number of cycles required to reach the light in this experiment and that required to rich a high performance in the experiments of Figure 4 is easily explained if you think that the correct behavior is more frequent than the wrong one as soon as performance is higher than 50%. The AutonoMouse starts therefore to approach the light source much before it has reached a high frequency of correct moves. In this and in the following experiments, 100 cycles were run in about 60 seconds.

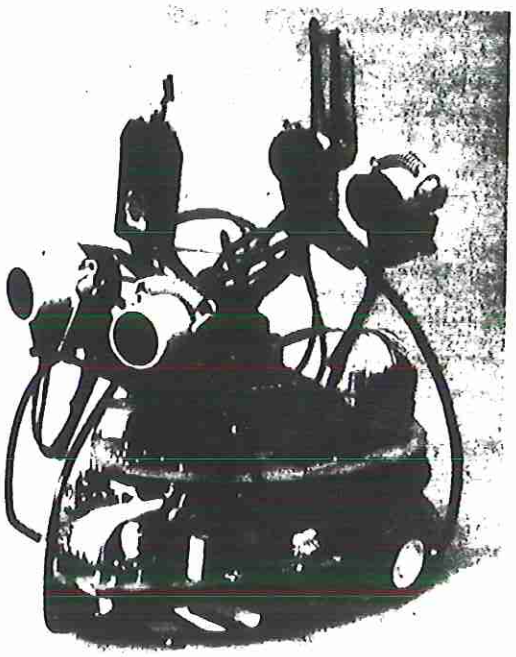

Figure 5. The AutonoMouse

Figure 7 shows the result of the same experiment after we inverted the AutonoMouse's eyes. (We made "hardware surgery" on the robot, connecting the previous right eye input to the left eye and vice versa.) Also in this experiment the light was moved away as soon as it was reached by the AutonoMouse (at cycle 150). The results of the experiments are qualitatively similar and show that the AutonoMouse was capable of adapting to the new situation with inverted eyes.

The comparison between the graphs of Figures 6 and 7 brings in the problem of the instrument to use to compare performance. In fact, as an experiment depends on variables like the randomly generated initial knowledge-base of the robot, the stochastic processes guiding the learning algorithms and, most important, the dynamics of the real environment, it cannot be exactly replicated. This means that, even if we make repetitive experiments with the same initial knowledge base and with equal pseudo-random sequences of numbers for the stochastic algorithms, still we would get different results. So, we are forced to give qualitative assessments of the results obtained, especially when comparing different runs. Most of the time, it turns out that it is very easy for a human observer to judge the quality of the results of an experiment through videorecordings; however, it is much more difficult to translate this qualitative assessment into numbers (or graphs). For the same reasons explained before, it is not interesting or not even possible to present "average graphs", i.e. graphs of results averaged on a number of experiments. All our graphs are relative to a single run. Different runs generated qualitatively similar graphs.

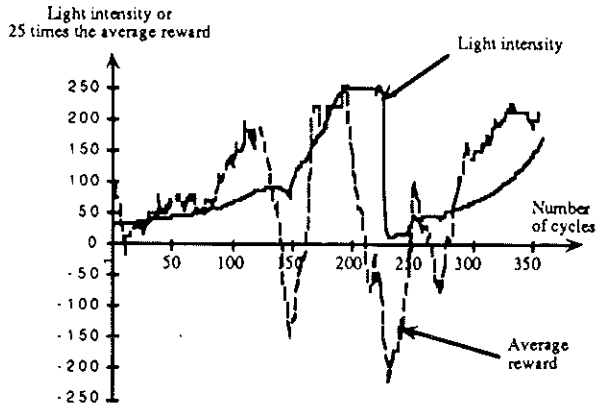


Figure 6. The robot learns to approach a stationary light source.

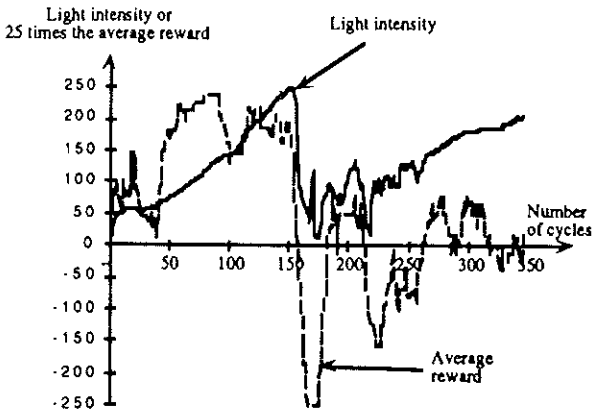


Figure 7. The robot learns to approach a stationary light source using inverted eyes.

In Figure 8 we show the result of an experiment in which one eye (the left one) of the AutonoMouse was disconnected. Again, the AutonoMouse was capable of reaching the light source, although with a somewhat degraded performance: it took 300 cycles to reach the light instead of the 150-200 of previous experiments. The drop in performance observed at cycle 135 is due to the fact that the AutonoMouse lost sight of the light and turned right until it saw the light source again.

Figure 9 shows the results of the last experiment of this paper. Here the task was to learn to follow a moving light source. In this experiment we faced a major dilemma; the reward function cannot use distance (or light intensity) changes to decide whether or not to give a reward. In fact, there are situations in which the AutonoMouse goes towards the light source, but at the same time the light source moves in the same direction. If the light source speed is higher than the AutonoMouse's, then the distance increases although the AutonoMouse made the right move. A possible solution is to let the AutonoMouse learn to approach a stationary light, then to freeze the rule set (i.e., stop the learning algorithm) and use it to follow the light. With this procedure one should be aware that, after learning has been stopped, the AutonoMouse can use only rules developed during the learning phase; this must therefore be as complete as possible. To ensure complete learning we need to give the AutonoMouse the opportunity to learn the right behavior in every possible environmental situation (in our case, every possible relative position of the light source).

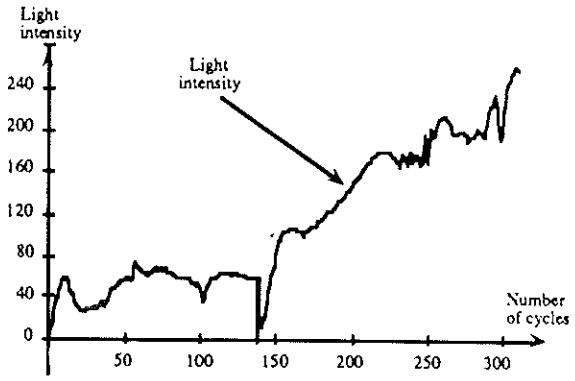


Figure 8. The robot learns to approach a stationary light source using only one eye (blind robot).

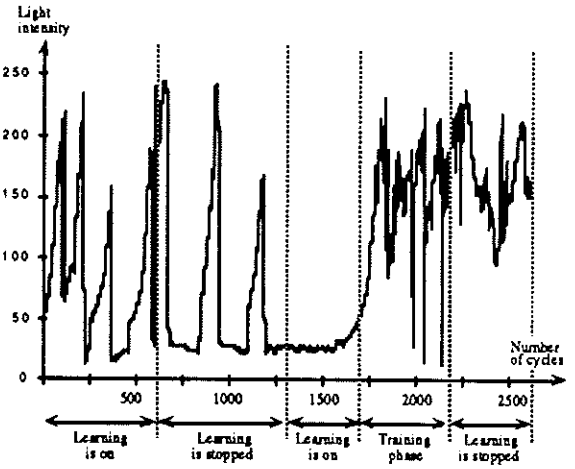


Figure 9. The robot learns to follow a moving light source.

In Figure 9 we report the results of the following experiment. At first the AutonoMouse is left free to move and to learn the approaching behavior. After 600 cycles we stop the learning algorithm, start to move the light source and let the AutonoMouse follow it. It appears that during the first learning phase the AutonoMouse did not learn all the relevant rules (in particular: it did not learn to turn right when the light was seen by the right eye) and therefore the resulting performance was not satisfactory. At cycle 1300 we therefore started the learning algorithm again and

performed a new training phase from cycle 1700 to 2200. During this training phase we presented the lamp to the AutonoMouse on one side, and waited until the AutonoMouse started to do the right thing. This procedure was repeated many times, presenting the light alternately in front, on the right, on the left and behind, until direct observation told us that ALECSYS had learned (i.e., the AutonoMouse was following the light). After this training phase, learning was stopped again (at cycle 2200) and the light source was steadily moved. Figure 9 shows that this time the observed behavior was much better (performance is far from maximum, however, because the light source is moving and the AutonoMouse therefore never reaches it).

## 6. Conclusions

In this paper we report a few results of an experimental activity in robot learning, carried out with the ALECSYS system in both the real and simulated worlds. The main goal of our research is to develop an autonomous robot through learning; in particular, we are interested in the use of learning classifier systems endowed with a hierarchical architecture.

Reinforcement learning appears to be suitable for shaping robots. Its main virtue is that the trainer need not present examples of correct behavior, and can rely on the robot's own exploration activity. As soon as the learning task becomes nontrivial, unfortunately, known learning algorithms tend to become slow and cumbersome. Naturally, we can hope to discover more powerful algorithms in the future; but many past experiences, like the ones of traditional Artificial Intelligence, warn us that magic algorithm may well not exist. In our opinion, the key issue for scaling up learning applications is "innate" architecture, that is the built-in structure of the learning agent.

Another problem with reinforcement learning is that the feedback information that guides learning is strongly local. This means that many behaviors are difficult to describe in terms of rewards and punishments. Take for example the light-following behavior, intuitively defined as "keep as close as possible to the moving light". In fact, such a description implicitly refers to a *global* property of the robot's behavior, which could be made explicit, for example, as the mean-squared distance of the robot from the light over a generic interval of time. However, to be translated into a program that gives rewards and punishments, the description of the desired behavior has to be transformed into a *local* criterion, for example: "reward the robot if it gets closer to the light, punish it if it gets farther". But this translation does not completely capture the original description.

We have a similar problem when we try to measure the robot's performance. Clearly, to provide sensible results such a measure must be strictly coupled with the reward program: the responses that are positively rewarded must count as good performance, and the ones that are punished must count as bad performance. However, it has constantly been clear to us during experiments that there is more in the robot's behavior than what can be plotted in a graph. In other words, there are *qualitative* aspects emerging from the robot's behavior, that are not described by performance graphs.

It appears that we need to set up the conceptual apparatus and the technical terminology of a new discipline, that might well be dubbed *ethology of the artificial*, in order to describe the qualitative aspects of emerging behavior of artificial agents. Such a discipline would be concerned with the classification and description of different kinds of behavior and environments, with the design of experimental protocols, and with the choice of reliable criteria for evaluating performance.

## Acknowledgements

This research was supported in part by a grant from CNR - Progetto finalizzato sistemi informatici e calcolo parallelo - Sottoprogetto 2 - Tema: Processori dedicati and from CNR - Progetto finalizzato robotica - Sottoobiettivo 2 - Tema: ALPI.

## References

Barto, A. G., R. S. Sutton & C. W. Anderson, 1983. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, **13**, 834-846.

Booker, L., 1988. Classifier Systems that Learn Internal World Models. Machine Learning, 3, 2-3, 161–192.

Brooks, R. A., 1991a. Intelligence without representation. *Artificial Intelligence*, **47**, 1-3, 139–159.

Brooks, R. A., 1991b. Artificial Life and Real Robots. *Proceedings of the 1st European Conference on Artificial Life (ECAL)*, Elsevier Publisher, Paris, France.

Brooks, R. A., 1992. Artificial life outside the computer. *Presented at Artificial Life III*, Santa Fe, NM.

Colombetti, M., & M. Dorigo, 1992. Robot Shaping: Developing Situated Agents through Learning. Technical Report 92-040, International Computer Science Institute, Berkeley, CA.

Dorigo, M., 1992a. *Optimization, learning and natural algorithms*, Ph.D. Thesis, Politecnico di Milano, Milano, Italy.

Dorigo, M., 1992b. ALECSYS and the AutonoMouse: Learning to control a real robot by distributed classifier systems. Technical Report 92–011, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy.

Dorigo, M., 1992c. Using Transputers to Increase Speed and Flexibility of Genetics-based Machine Learning Systems. *Microprocessing and Microprogramming Journal*, 34, 147-152.

Dorigo, M., & U. Schnepf, 1993. Genetics-based machine learning and behavior-based robotics: A New Synthesis. *IEEE Transactions on Systems, Man, and Cybernetics*, **23**, 1.

Dorigo, M., & E. Sirtori, 1991. ALECSYS: A parallel laboratory for learning Classifier systems, *Proceedings*

*of Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Diego, CA.

Holland, J. H., & J. S. Reitman, 1978. Cognitive systems based on adaptive algorithms. In D.A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems.* Academic Press, New York.

Lin, L-J., 1991. Programming robots using reinforcement learning and teaching. *Proceedings of the Ninth National Conference on Artificial Intelligence, AAAI-91*, 781-786.

Lin, L-J., 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, **8**, 3, 293-322.

Mahadevan, S., 1992. Enhancing transfer in reinforcement learning by building stochastic models of robots actions. *To appear in the Proceedings of the Ninth Conference on Machine Learning* , Aberdeen, Scotland.

Mahadevan, S., & J. Connell, 1992. Automatic programming of behavior-based robots using reinforcement learning, *Artificial Intelligence*, **55**, 2.

Robertson, G.G., & R. L. Riolo, 1988. A tale of two classifier systems. *Machine Learning*, **3**, 2-3, 139–160.

Singh, S., 1992a. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, **8**, 3/4, 323–339.

Singh S., 1992b. Reinforcement learning with a hierarchy of abstract models. *Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI-92*, 202–207.

Sutton, R.S., 1984. Temporal credit assignment in reinforcement learning. Ph.D. dissertation, Dep. of computer and information science, University of Massachusetts, Amherst, MA.

Tinbergen, N., 1966. *The Study of Instincts.* Oxford University Press, 1966.

Watkins, C. J. C. H., 1989. *Learning with delayed rewards.* Ph.D. dissertation, Psychology Department, University of Cambridge, England.

Watkins, C. J. C. H., & P. Dayan, 1992. Technical Note: Q-learning. *Machine Learning*, **8**, 3-4, 279–292.

Williams, R. J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, **8**, 3-4, 229–256.

Wilson, S., 1987. Classifier systems and the Animat problem. *Machine Learning*, **2**, 3, 199–228.

# FROM ANIMALS TO ANIMATS 2

## Proceedings of the Second International Conference on Simulation of Adaptive Behavior

edited by Jean-Arcady Meyer, Herbert L. Roitblat, and Stewart W. Wilson


```
@incollection{ColDor1992:sab,
      Address = {Cambridge, MA},
      Author = {M. Colombetti and M. Dorigo},
      Booktitle = {From Animals to Animats 2:
                   Proceedings of the Second International Conference on
                   Simulation of Adaptive Behavior},
      Editor = {J. A. Meyer and H. L. Roitblat and S. W. Wilson},
      Pages = {305--312},
      Publisher = {MIT Press},
      Title = {Learning to Control an Autonomous Robot by Distributed Genetic Algorithms},
      Year = {1992}
      }
```

A Bradford Book

The MIT Press
Cambridge, Massachusetts
London, England