Evolving non-Trivial Behaviors on Real Robots: an Autonomous Robot that Picks up Objects

Stefano Nolfi **Domenico Parisi**

Institute of Psychology, National Research Council
15, Viale Marx - 00187 - Rome - Italy
voice: 0039-6-86090231
stefano@kant.irmkant.rm.cnr.it
domenico@kant.irmkant.rm.cnr.it

Abstract

Recently, a new approach that involves a form of simulated evolution has been proposed for the building of autonomous robots. However, it is still not clear if this approach may be adequate to face real life problems. In this paper we show how control systems that perform a non-trivial sequence of behaviors can be obtained with this methodology by carefully designing the conditions in which the evolutionary process operates. In the experiment described in the paper, a mobile robot is trained to locate, recognize, and grasp a target object. The controller of the robot has been evolved in simulation and then downloaded and tested on the real robot.

1 Introduction

Work in *Artificial Life* (see Langton, 1989) has introduced new techniques for developing creatures living and behaving in a variety of environments. More recently, there have been several attempts to apply Artificial Life techniques to the design of mobile robots. This type of approaches, identified as *Evolutionary Robotics* because they try to develop autonomous robots through an automatic design process involving artificial evolution, have attracted the interest of researchers of both the Artificial Life and the Robotics communities (Brooks, 1992; Cliff, Husband and Harvey, 1993; Nolfi, Floreano, Miglino, and Mondada, 1994; Steels, 1994; and others).

Evolutionary Robotics approaches are based on the genetic algorithm technique (Holland, 1975). An initial population of different "genotypes" each codifying the control system (and possibly the morphology) of a robot are created randomly. Each robot is evaluated in the environment and to each robot is assigned a score ("fitness") corresponding to the ability of the robot to perform some desired task. Then, the robots that have obtained the highest fitness are allowed to reproduce (sexually or agamically) by generating copies of their genotypes with the addition of random changes ("mutations"). The process is repeated for a certain number of generations until, hopefully, desired performances are achieved (for methodological information see Nolfi, Floreano, Miglino and Mondada, 1994).

2 Related Work

Different types of artificial systems that perform various behaviors have been obtained through artificial evolution. However, the majority of these systems have been obtained and tested in simulations without being validated on real robots. Although these simulated models can be useful for exploring many theoretical and practical questions, care must be taken in using them to draw conclusions about behavior in real world.

Only recently the evolutionary approach has produced results that have been validated on real robots. Lewis, Fagg and Sodium (1992) evolved a motor controller for a six-legged robot called Rodney that was able to walk forward and backward. Colombetti and Dorigo (1992) have evolved a the control system for a robot called Autonomouse to perform a light approaching and following behavior. Floreano and Mondada (1994) and Nolfi, Floreano, Miglino, and Mondada (1994) have evolved control

systems for the miniature mobile robot called Khepera (see below) that should perform an obstacle avoidance task. Miglino, Nafasi, and Taylor (in press) evolved a controller for a mobile Lego robot that should explore an open arena. Yamauchi and Beer (1994) describe an experiment in which dynamical neural networks were evolved to solve a landmark recognition task using a sonar. They tested the network on a Nomad 200 robot with a built in wall-following behavior. Harvey, Husband, and Cliff (1994) evolved a system able to approach a visual target (in the most complex experiment the system was successfully trained to approach a triangle target and to distinguish it from a rectangular one). The system was not implemented on a standard autonomous robot but on a specially designed robotic equipment in which the robot is suspended from a platform which allows translational movements in the X and Y directions.

In some of these experiments the evolutionary process was conducted in simulation and then the obtained control system was downloaded and tested on the robot (Colombetti and Dorigo, 1992; Miglino, Nafasi, and Taylor, in press; Yamauchi and Beer, 1994). In other cases the evolutionary process was conducted entirely on the real robot (Lewis, Fagg, and Sodium, 1992; Floreano and Mondada, 1994, Harvey, Husband, and Cliff, 1994). In still other cases evolution took place in part in simulation and then it was continued on the real robot (Nolfi, Floreano, Miglino, and Mondada, 1994). When the evolutionary process was conducted partially or totally on the real robot in some cases the evaluation process was conducted automatically, i.e., without requiring an external support (Floreano and Mondada, 1994; Nolfi, Floreano, Miglino, and Mondada, 1994) while in other cases performance were evaluated by human observers (Lewis, Fagg and Sodium, 1992). In all of the work described neural networks were used in order to implement the controller with the exception of Colombetti and Dorigo (Colombetti and Dorigo, 1992) who used a classifier system.

This work clearly shows that evolutionary robotics is a very active and promising new field of research. However, it is also clear that real life problems require system able to produce behaviors significantly more complex than those described above. In this paper we will show how control systems that perform a more complex sequence of behaviors can be obtained with this methodology by carefully designing the conditions in which the evolutionary process operate. In doing so we will also discuss several methodological issues.

3 Our Framework

Having at our disposal a Khepera robot with the gripper module (see next section) we decided to try to develop a control system for a robot that, when placed in an arena surrounded by walls is able to recognize target objects, to grasp them, and to carry them outside the arena. Given our prevous experience with Khepera and the difficulties of evolving a behavior of this type in the real robot we decided to conduct the evolutionary process in simulation by using an extended version of our Khepera simulator (described in Nolfi, Floreano, Miglino, and Mondada, 1994). The obtained control system was then downloaded on the robot and tested in the real environment. In this section we will describe the robot, the environment of the robot, and the simulator. In section 4 we will describe the architecture of the controller and the type of genetic algorithm and fitness formula used. In section 5 we will describe the results obtained in the simulations and on the real robot.

3.1 The Robot

The robot was Khepera (Figure 1), a miniature mobile robot developed at E.P.F.L. in Lausanne, Switzerland (Mondada, Franzi, and Ienne, 1993). It has a circular shape with a diameter of 55 mm, a height of 30 mm, and a weight of 70g. It is supported by two wheels and two small teflon balls. The wheels are controlled by two DC motors with an incremental encoder (10 pulses per mm of advancement of the robot), and they can move in both directions. In addition, the robot is provided with a gripper module with two degrees of freedom. The arm of the gripper can move any angle from vertical to hotizontal while the gripper can assume only the open or closed position. The robot is provided with eight infra-red proximity sensors (six sensors are positioned on the front of the robot, the remaining two on the back), a speed of rotation sensor for each motor, an optical barrier sensor on the

gripper able to detect the presence of an object in the gripper, and an electrical resistivity sensor also on the gripper.

A Motorola 68331 controller with 256 Kbytes of RAM and 512 Kbytes ROM manages all the input-output routines and can communicate via a serial port with a host computer. Khepera was attached to the host by means of a lightweight aerial cable and specially designed rotating contacts. This configuration makes it possible to trace and record all important variables by exploiting the storage capabilities of the host computer and at the same time it provides electrical power without using time-consuming homing algorithms or large heavy-duty batteries.

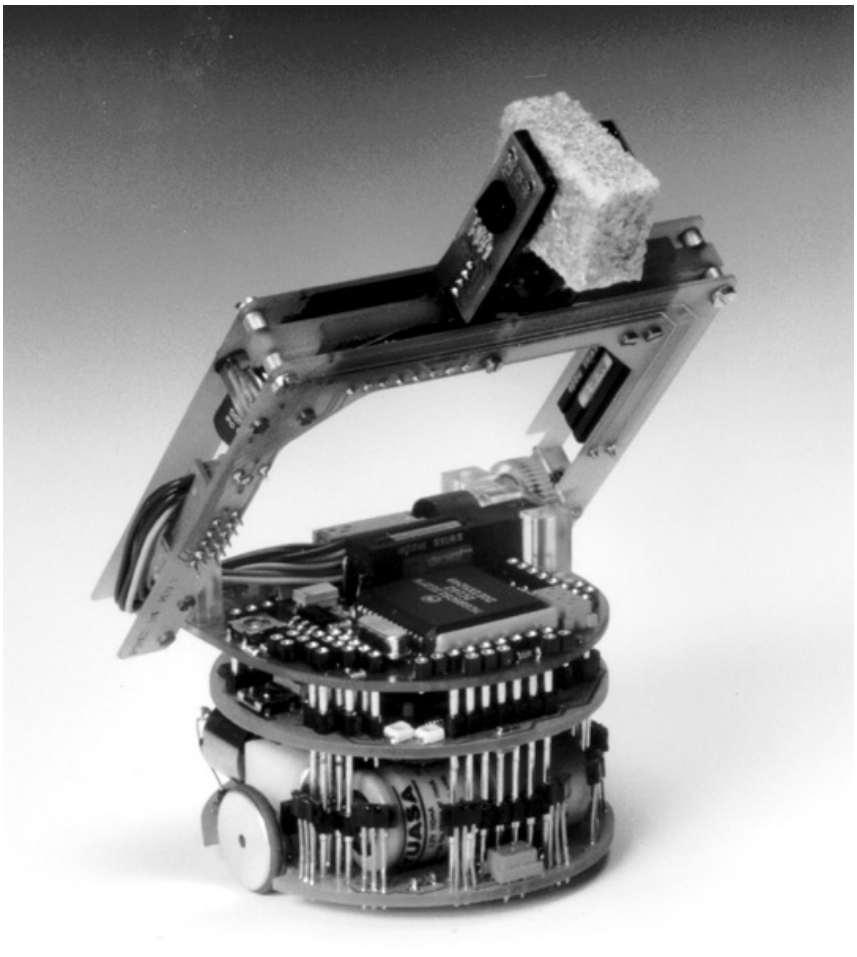

Fig.1. The Khepera robot.

3.2 The Environment

We built a rectangular environment of 60x35 cm organized as an arena surrounded by walls. The walls are 3 cm in height, are made of wood, and are covered with white paper. The target object is a cilinder with a diameter of 2.3 cm and a height of 3 cm. It is made of cardboard and is covered with white paper. The target is positioned in a random position inside the arena.

3.3 The Simulator

To evolve the controller of the robot in the computer the simulator described in Nolfi, Floreano, Miglino, and Mondada (1994) was extended in order to take into account the gripper module of Khepera.

A sampling procedure is been used to calculate values for the infra-red sensors. The walls and the target object were sampled by placing the physical Khepera in front of them, letting it turn 360^{o}, and at the same time recording the sensory activations at different distances with respect to objects. The activation level of each of the eight infra-red sensors was recorded for 180 different orientations and for 20 different distances. In this way two different matrix of activations where obtained for the two types of objects (walls and target). These matrices were then used by the simulator to set the activation state of the simulated sensors depending of the relative position of Khepera and of the objects in the simulated environment. (When more than one object is within the range of activation of the sensors, the resulting activation was computed by summing the activation contribution of each object). This sampling procedure may result time consuming in the case of very unstructured environments because it requires to sample each different type of objects present in the environment. However, it has the advantage of taking into account the fact that different sensors, even if identical from the electronic point of view, do respond differently. Sampling the environment throught the real sensors of the robot allowed us, by taking into account the characteristics of each individual sensor, to develop a simulator shaped by the actual physical characteristics of the individual robot we have.

The effects of the two motors were sampled similarly by measuring how Khepera moved and turned for a subset of the 20x20 possible states of the two motors. At the end of this process a matrix was obtained that was then used by the simulator in order to compute the displacements of the robot in the simulated environment.

The physical shape of Khepera (including the arm and the gripper), the environment structure, and the actual position of the robot were accurately reproduced in the simulator with floating point precision. Motor actions that may produce a crashing of the robot into the walls are not executed in the simulated environment. Therefore, the robot may get stuck into the walls if it is unable to avoid them. On the contrary, crashes between the arm and the target object are simulated by re-positioning the target object in a new randomly selected position inside the arena.

4 Evolving the Controller

Like the majority of people who use evolutionary methods to obtain control systems for autonomous robot we decided to implement the controller with a neural network. This was motivated by several reasons:

- Neural networks are resistant to noise that is massively present in robot/environment interactions.
- We agree with Cliff, Harvey, and Husband (1993) that the primitives manipulated by the evolutionary process should be at the lowest possible level in order to avoid undesiderable choices made by the human designer. Synaptic weights and nodes are low level primitives.
- Neural networks can easily exploit various form of learning during life-time and this learning process may help and speed up the evolutionary process (Ackley and Litmann, 1991; Nolfi, Elman and Parisi; 1994).

In the following sections we will describe the architecture, the fitness formula, and the form of genetic algorithm used.

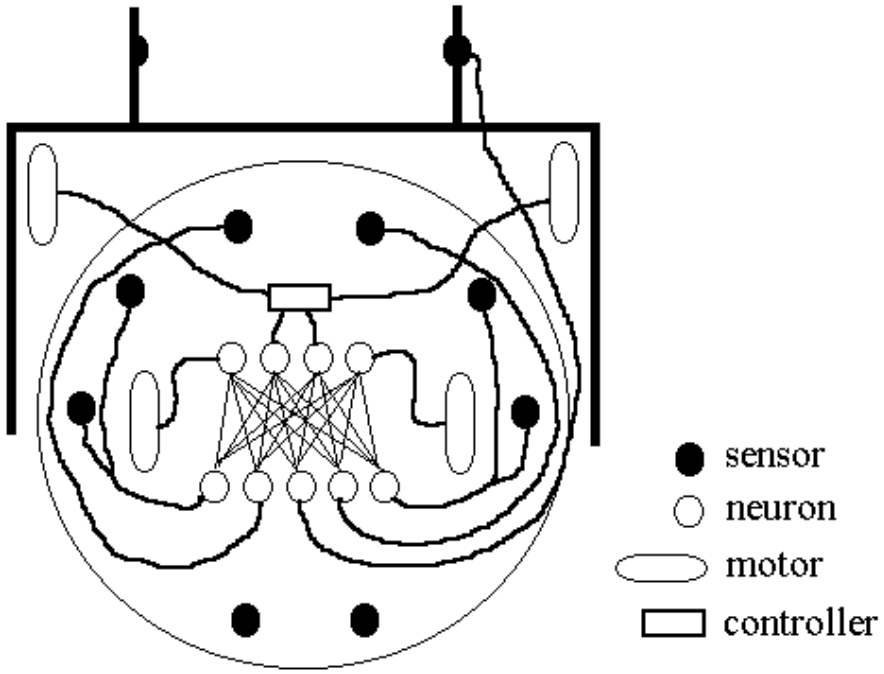

Fig. 2. The control system of the robot. The 5 sensory neurons are directly connected to the 6 frontal sensors of the robot located on the body of Khepera and to the light barrier sensor located on the gripper. Two motor neurons are directly connected with the two wheels of Khepera while the other two neurons are connected with the two motors that control the arm and the gripper.

4.1 The Neural Controller

We tried several different network architectures (we will come back to this point in the discussion) and we found that the best architecture was a very simple one, a feedforward network with 5 sensory neurons, 4 motor neurons, and no internal neurons. The first 4 sensory neurons were used to encode the activation level of the two frontal sensors of Khepera and the average activation of the two left and right lateral sensors (see Figure 2). The fifth sensory neuron was used to encode the barrier light sensor on the gripper. On the motor side, the first 2 neurons were used to encode the movement of the two wheels, the third neuron was used to trigger the object pick-up procedure, and the last neuron was used to trigger the object release procedure.

The activation of the sensors and the state of the motors are encoded each 100 milliseconds. However, when the activation level of the object pick-up neuron or of the object release neuron reach a given treshold a sequence of action occurs that may require one or two seconds to complete (e.g. move a little bit back, close the gripper, move the arm up, for the object pick-up procedure; move the arm down, open the gripper, and move the arm up again, for the object release procedure). This implies that in order to accomplish the task the weights of the neural network should be set in a way that allows Khepera to perform the following sequence of behaviors:

- explore the environment avoiding the walls
- recognize the target object
- place the body in a relative position with respect to the target that makes it possibile to grasp the object
- pick-up the target object
- move toward the walls without avoiding them
- place the body in a relative position with respect to the wall that makes it possible to drop the object out of the arena when released
- release the object

4.2 The Genetic Algorithm

To evolve neural controllers able to perform the task described above we used a form of genetic algorithm. We begin with 100 randomly generated genotypes each representing a network with a different set of connection weights assigned randomly. This is Generation 0 (G0). G0 networks are allowed to "live" for 5 epochs, with an epoch consisting of 500 actions (about 5 seconds in the simulated environment using an IBM RISK/6000 computer or about 250 seconds in the real environment). At the beginning of each epoch the robot and the target object are randomly positioned in the arena and during each epoch the object is re-positioned randomly when the robot releases the object after having picked it up. At the end of life the robots have the possibility to reproduce. However, only the 20 individuals which have accumulated the most fitness in the course of their life reproduce (agamically) by generating 5 copies of their neural networks. These 20x5=100 new robots constitute the next generation (G1). Random mutations are introduced in the copying process resulting in possible changes of the connection weights. The process is repeated for 300 generations.

The genetic encoding scheme was a direct one-to-one mapping. The encoding scheme is the way in which the phenotype (in this case the connection weights of the neural network) is encoded in the genotype (the representation on which the genetic algorithm operates). The one-to-one mapping is the simplest encoding scheme in which to each phenotypical character corresponds one and only one 'gene'. In our case to each connection weights corresponds a sequence of 8 bits in the genotype which has a total length of 192 bits. (For more complex encoding schemes that allow the evolution of the neural architecture, see Cliff, Harvey and Husband, 1993; Nolfi, Miglino, and Parisi, 1994).

4.3 The Fitness Formula

The fitness formula is the way in which individuals are evaluated in order to decide which individuals are allowed to reproduce. To help the evolutionary process we decided to use a fitness formula with 5 components in order to score individuals not only for their ability to perform the complete sequence of correct behaviors but also for their ability to perform only portions of the complete sequence. In particular, we increased the fitness of an individual in the following cases:

- if the individual is close to the target object
- if the target object is in front of the robot
- if the robot tries to pick-up the object
- if the robot has the object in the gripper
- if the robot release the object outside the arena

It is important to note that, despite the complexity of the fitness formula, several behaviors that are necessary in order to accomplish the task are not directly rewarded. For example, the ability to avoid the walls, the ability to explore the environment efficiently in order to find the target object, and the ability to distinguish between the walls and the target object, are not directly rewarded. Similarly, there is no direct reward for correct or incorrect behaviors after the object has been grasped. For example, if the robot decides to stay still or to move makes no difference for the obtained fitness. Only if the robot performs the entire sequence of correct behaviors ((a) moving in the direction of a wall, (b) approaching the wall instead of avoiding it as before grasping the object, (c) releasing the object, it is rewarded.

5 Results

We run 10 simulations starting with populations of 100 networks with randomly assigned connection weights. Each simulation lasted 300 generations (about 3 hours using a standard IBM RISC/6000). In all 10 simulations individuals able to perform the task rather well evolved.

We tested the best individual of the last generation for each simulation. On average, in the simulated evironment, individuals were able to perform the complete sequence of behaviors 6.6 times (i.e., in the

500x5 cycles of their life they were able to find, recognize, pick up, transport, and correctly release outside the arena 6.6 target objects on average). Incorrect behaviors (such as crashing into the walls) were generated by the evolved individuals very unfrequently. In fact, individuals never tried to grasp a wall, failed to grasp the object only 2% of the times along 500x5 cycles, and released the object in an incorrect position (i.e., inside the arena) only 10% of the times.

We then downloaded the 10 evolved networks into the physical robot and we tested them in the real environment. The performance of the 10 best individuals resulted less good, on average, in the real than in the simulated environment. However, all individuals were able to perform the entire sequence of actions correctly at least 2 times. The best individual, out of the ten, was able to correctly pick-up and deposit outside the arena 6 target objects, it never crashed into the walls, it never failed to grasp the objects, and it never tried to incorrectly grasp the walls. Given this successful performance it did not seemed necessary to continue the evolutionary process in the real environment in order to allow individuals to adapt to the differences between the simulated and the real environment, as we did in our previous work (see Nolfi, Floreano, Miglino, and Mondada, 1994).

6 Discussion

We were able to evolve neural controllers for a Khepera robot that can perform a relatively complex task. Hence, one first conclusion we can draw from this work it that the evolutionary robotics approach appears to be adequate to face real life problems in simple environments.

The present work can also help us to find an answer to the question: In what conditions is is possible to evolve robots that are able to perform complex behaviors?

We think the key point is to view robots and their environments "as agent-environment systems whose interaction dinamics have to be got right" instead of "thinking of robots as information processing systems and of sensors as measuring device" (Smithers, 1994; see also Nolfi, Floreano, Miglino, and Mondada, 1994). In other words, what is important is to reduce the complexity of the interaction between each component of the system (body, controller, sensory and motor system) and the given environment. And this should be accomplished by designing each component by taking into account all the others, i.e., by adapting each component to each others, instead of trying to design very smart and complex components that are general purpose.

The type of neural controllers we evolved and the hardware we used are very simple. The evolved controllers, in fact, are implemented on neural networks with an extremely simple architecture and with only 22 free parameters to optimize (20 weights values and 2 biases). On the body side, the infrared sensors used are both very imprecise and very noisy devices. In addition we found that by increasing the complexity of the system on the control side (by using more complex neural architecture, such as architectures with internal or recurrent units) or on the body side (by allowing the neural network to rely on more sensors) makes the task harder rather than easier to evolve and does not produce better performances.

To design by hand each component of the system by taking into account all the others is certainly difficult. For this reason an auto-organization process such as evolution that spontaneously allows co-adaptation of the sub-components of the systems appears a good solution to develop simple robots that perform complex tasks. In this work, in order to reduce the number of parameters to be optimized, we applied the evolutionary process only to the weights of the neural controller and we tried to carefully design the neural architecture and the sensory-motor system. However, the auro-organization process can be extended to the neural architecture and the sensory system and this may be a better solution in principle and a necessary one in complex cases.

A final remark concerns the fitness formula we used. As we noted in section 4.3, not all required sub-behaviors necessary to accomplish the task were directly rewarded by the fitness formula. However, some of the intermediate states prior to the final goal state were directly rewarded. It is clear that to

reward only the desired final state (the object is out of the arena) instead of also some of the intermediate states would eliminate constraints, on the evolutionary process, that have been introduced by hand and that therefore may result inadequate. We decide to reward some of the intermediate states because it appeared necessary given the difficulty of the task. However, it is not clear if there are other ways to solve this type of problem. We are presently trying to determine if and to what extent it is necessary to impose this type of constraints in the case of the task desrcibed in this paper.

Acknowledgments

This research has been supported by P.F. "ROBOTICA", National Research Council, Italy.

References

Ackley, D. H., M. L. Littman, 1991. Interactions between learning and evolution. In *Artificial Life II*, edited by C. G. Langton, J. D. Farmer, S. Rasmussen, C. E. Taylor. Reading, Mass., Addison-Wesley.

Brooks, R. A. 1992. Artificial life and real robots. *In Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life,* edited by F. J. Varela, P. Bourgine. Cambridge, Mass, MIT Press/Bradford Books.

Cliff D. T., I. Harvey, P. Husbands. 1993. Explorations in Evolutionary Robotics. *Adaptive Behavior* **2**: 73-110.

Colombetti M., Dorigo M. 1992. Learning to control an autonomous robot by distributed genetic algorithms. In J. A. Meyer, H. L. Roitblat, and S. W. Wilson, (eds), *From Animals to Animats 2*, Proceedings of 2rt International Conference on Simulation of Adaptive Behavior, MIT Press.

Floreano D., F. Mondada. 1994. Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. In *From Animals to Animats 3: Proceedings of Third Conference on Simulation of Adaptive Behavior,* edited by D. Cliff, P. Husbands, J. Meyer, S. W. Wilson. Cambridge, Mass, MIT Press/Bradford Books.

Harvey I., Husband I., Cliff, D. 1994. Seeing the light: artificial evolution, real vision. In D. Cliff, P. Husband, J-A Meyer, and S. Wilson, (eds), *From Animals to Animats 3*, Proceedings of 3rt International Conference on Simulation of Adaptive Behavior, MIT Press/Bradford Books.

Holland J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor, Mich., University of Michigan Press.

Langton, C. G. 1989. *Proceedings of Artificial Life*. C.G. Langton (ed.), Addison-Wesley.

Lewis, M.A., Fagg, A.H., Sodium, A. 1992. Genetic programming approach to the contruction of a neural network for control of a walking robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France.

Miglino O., K. Nafasi, C. Taylor. in press. Selection for Wandering Behavior in a Small Robot. *Artificial Life*.

Mondada F., E. Franzi, P. Ienne. 1993. Mobile Robot miniaturisation: A tool for investigation in control algorithms. In: *Proceedings of the Third International Symposium on Experimental Robotics*, Kyoto, Japan.

Nolfi, S., Miglino, O., Parisi, 1994. Phenotypic Plasticity in Evolving Neural Networks. In: D. P. Gaussier and J-D. Nicoud (Eds.) Proceedings of the Intl. Conf. From Perception to Action, Los Alamitos, CA: IEEE Press

Nolfi, S., Florano D., Miglino, O., Mondada, F. 1994. *How to evolve autonomous robots: different approaches in evolutionary robotics*. Proceedings of fourth International Conference on Artificial Life, Cambridge MA, MIT Press.

Nolfi, S., Elman, J.L., Parisi, D. 1994. Learning and Evolution in Neural Networks. *Adaptive Behavior*,vol. 3, **1**, pp.5-28.

Smithers T. 1994. On why better robots make it harder. In D. Cliff, P. Husband, J-A Meyer, and S. Wilson, (eds), *From Animals to Animats 3*, Proceedings of 3rt International Conference on Simulation of Adaptive Behavior, MIT Press/Bradford Books.

Steels L. 1994. *Emergent functionality in robotic agents throught on-line evolution*. Proceedings of fourth International Conference on Artificial Life, MIT Press, Cambridge MA.

Yamauchi B., Beer R. 1994. Integrating reactive, sequential, and learning behavior using dinamical neural networks. In D. Cliff, P. Husband, J-A Meyer, and S. Wilson, (eds), *From Animals to Animats 3*, Proceedings of 3rt International Conference on Simulation of Adaptive Behavior, MIT Press/Bradford Books.