Institute of Psychology
C.N.R. - Rome

Using emergent modularity to develop control systems for mobile robots

Stefano Nolfi
Institute of Psychology, National Research Council, Rome, Italy.
e-mail: stefano@kant.irmkant.rm.cnr.it

June 1996 (revised December 1996)

Technical Report 96-14

Department of Neural Systems and Artificial Life
15, Viale Marx
00137 - Rome - Italy
voice: 0039-6-86090231
fax: 0039-6-824737

To appear on: *Adaptive Behavior*, Special issue on "Complete agent learning in complex environment"

Using emergent modularity to develop control systems for mobile robots

Stefano Nolfi
Institute of Psychology, National Research Council
15, Viale Marx - 00187 - Rome - Italy
voice: 0039-6-86090231
fax: 0039-6-824737
stefano@kant.irmkant.rm.cnr.it

Abstract

A new way of building control systems, known as behavior based robotics, has recently been proposed to overcome the difficulties of the traditional AI approach to robotics. This new approach is based upon the idea of providing the robot with a range of simple behaviors and letting the environment determine which behavior should have control at any given time. We will present a set of experiments in which neural networks with different architectures have been trained to control a mobile robot designed to keep an arena clear by picking-up trash objects and releasing them outside the arena. Controller weights are selected using a form of genetic algorithm and do not change during the lifetime (i.e. no learning occurs). We will compare, in simulation and on a real robot, five different network architectures and will show that a network which allows for fine-grained modularity achieves significantly better performance. By comparing the functionality of each network module and its interaction with a description of the simple behavior components, we will show that it is not possible to find simple correlations; rather, module switching and interaction is correlated with low-level sensory-motor mappings. This implies that the engineering-oriented approach to behavior-based robotics might have serious limitations because it is difficult to know in advance the appropriate mappings between behavior components and sensory-motor activity for complex tasks.

1. Introduction

A new way of building control systems, known as *behavior based robotics*, has recently been proposed to overcome the difficulties of the traditional AI approach to robotics (Brooks, 1986). This new approach is based upon the idea of providing the robot with a range of simple behaviors and letting the environment determine which behavior should have control at any given time. Despite the central role of the environment however, behavior based systems differ from purely reactive systems because "they can use different forms of internal representations and perform computations on them in order to decide what effector action to take" (Mataric, 1992).

The design of a control system centered on the behavior based approach usually involves breaking down the required behavior into a set of basic behaviors (also called reflexes), such us "approach" or "avoid", which are specified from the observer's point of view, and designing an action selection or coordination mechanism able to ensure that only the correct basic behavior has the control over the actuators at the right time. Most of the time both the modules of the controller corresponding to the defined basic behaviors and the action selection mechanism are designed by the experimenter even if the design process is accomplished often incrementally and involves intensive testing and debugging. However, it has also been shown that basic behaviors and action selection can be learned (Maes 1992, Mahadevan and Connell, 1992; Dorigo and Schnepf, 1993).

We claim that, in order to really obtain simple and robust solutions, the process of breaking down the required behavior into sub-components and of integrating them should be accomplished taking into account the

"proximal" description of behavior (i.e. the sensory-motor loops responsible for the resulting behavior of an individual) and not the "distal" description of behavior (i.e. a high level in which terms such as "approach" or "avoid" are used to describe, from the observer's point of view, the result of a sequence of sensory-motor loops). This can be accomplished, as we will show in this paper, by using learning or adaptation not only to develop the module of the controller responsible for the basic behaviors and action selection, but also to break the required behavior down into basic behaviors to be coordinated/selected.

To investigate this issue we will present a set of experiments in which neural networks with different architectures were trained to control a mobile robot designed to keep an arena clear by picking-up trash objects and releasing them outside the arena. The obtained results, validated on the real robot, show how the best performances are obtained with an emergent modular architecture (i.e. an architecture in which the different modules responsible for the different sub-behaviors, the action selection mechanism, and the process of breaking down the required behavior into basic behaviors are the result of an adaptation process). Moreover, the analysis of the successfully trained individuals shows that there are no correspondence between neural modules and distal description of behaviors and that in order to understand the role of modularity one should look at the proximal description of behavior.

1.1 Describing behaviors of agents that interact with an external environment

The main goal of research in robotics is the attempt to provide a methodology for synthesizing control systems for physical robots able to produce complex behavior. However, the fact that simple mechanisms, in the appropriate environment, may exhibit complex behaviors (Braitenberg, 1984) raises the problem of defining more clearly the terms behavior and complexity (referred to behavior).

One way to overcome this problem is, as proposed by Sharkey and Heemskerk (in press), to distinguish two ways of describing behavior: a description from the point of view of the observer in which high level terms such as "approach" or "attack" are used to describe the result of a sequence of sensory-motor loops (*distal description of behavior*) and a description from the point of view of the agent's sensory-motor system that accounts for how the agent itself reacts in different sensory situations (*proximal description of behavior*). The distal description of a behavior is a function not only of the controller determining how the agent reacts to each possible sensory stimulus but also of the environment and of the agent's sensory and motor apparatus. As a consequence, simple controllers may be able to produce behaviors that, even if simple in a proximal description, may appear complex in a distal description.

Another important characteristic of the behavior of mobile agents is that by actively interacting with the external environment, for example by moving, they are able to partially determine the kind of stimuli they are exposed to (see Parisi, Cecconi, Nolfi, 1990). A response to a sensory state results in a new sensory state and in a new motor response that in its turn results in a new sensory state, forming a sensory-motor loop. This fact has several implications: (a) agents can produce behavioral sequences without using any form of memory; (b) agents can self-select favorable sensory states and avoid unfavorable ones (Nolfi and Parisi, 1993). This also implies that not all the sensory-motor states that constitute the proximal description of an agent's behavior have the same importance. Some sensory-motor states may be extremely important (for example because they allow the agent to avoid falling into an undesirable behavioral loop), while others may be completely irrelevant because they are never encountered (given the fact that the agents themselves determine the successive sensory states) or because, whatever motor response the agent produces, it in any case falls within a behavioral loop.

2. The experimental framework

Having at our disposal a Khepera robot with the gripper module (see below) we decided to try to develop a control system for a robot with the task of keeping clear an arena surrounded by walls. The robot should look for "garbage", somehow grasp it, and take it out of

the arena. The task of cleaning the arena can be broken down into several sub-tasks: (a) explore the environment, avoiding the walls; (b) recognize a target object and to place the body in a relative position so that it can be grasped; (c) pick up the target object; (d) move toward the walls while avoiding other target objects; (e) recognize a wall and place the body in a relative position that allows the object to be dropped out of the arena; (g) release the object. Moreover, these sub-tasks can be broken down into smaller components. For example (a) may be broken down into (a_1) go forward when sensors are not activated; (a_2) turn left at a given speed when right sensors are activated etc. However, as we want the complete solution to the task to emerge through an evolutionary process we do not need to specify the requested behavior in detail or to analyze the interference between basic behaviors.

Scheier and Pfeifer (1995) developed the control systems for a Khepera robot that performs a task very similar to the one described in this paper (see also Scheier and Lambrinos, 1995). The environment is an arena surrounded by walls which contains large and small pegs and a home base with a light source attached to it. The robot has to bring the small pegs to the home base. It was decided to design by hand a set of modules each corresponding to an elementary behavior (move forward, turn toward objects, avoid obstacles, grasp, and bring to the nest). All that is acquired during the training phase is the tuning of the grasp behavior. The robot is pre-programmed to turn around pegs and the size of the pegs determines the way in which the angular velocity of the robot changes in time. Reinforcement learning is used to associate the vectors of angular velocities corresponding to small pegs with grasping behavior; in other words to classify the two types of pegs. On the other hand, in this paper we want the entire control system, including its organization into modules corresponding to basic behaviors, to emerge during the training phase.

Colombetti, Dorigo, and Borghi (1996) also studied a task similar to that described in this paper. They trained a mobile robot based on a commercial platform produced by RoboSoft to collect food pieces and to store them in a nest. Each piece of food is a cylinder, wrapped in violet paper, which slides on to the floor when pushed by the robot. Nest position is marked by another cylinder wrapped in pink paper. The robot uses a frontal color camera to identify the position of food cylinders and of the nest, using colors to discriminate. Moreover, the nest sensor uses an odometer to get the approximate position of the nest when it is not visible.

To build the controller the authors decomposed the target behavior into a collection of simple behaviors (leave-nest, get-food, reach-nest, avoid-obstacles, coordinate-behaviors) and allocated a behavioral module to each of them (behavioral modules have been implemented using classifier systems). The behavioral modules (with the exception of the obstacle-avoidance module, which was pre-programmed) were trained separately and then frozen. The coordinator module was then trained to achieve the target behavior. However, they decided how to decompose the target behavior into basic behaviors while we want also this subdivision to be the result of a training phase. As Colombetti, Dorigo, and Borghi note at the end of their paper: “In order to relieve designers from part of their burden, learning techniques might be extended to other aspects of robot development, like the architecture of the controller. This means that the structure of behavioral modules should emerge from the learning process, instead of being pre-designed.” (Colombetti, Dorigo, and Borghi, 1996).

We decided to train controllers using an evolutionary method (Cliff, Harvey, and Husband, 1993; Nolfi, Floreano, Miglino, and Mondada, 1994; Mataric, and Cliff, in press) and to conduct the training process in simulation (for a description of the simulator see Miglino, Lund, and Nolfi, 1995). The control systems were then downloaded into the robot and tested in the real environment. In this section we will describe the robot and the environment, the architecture of the controller, and the genetic algorithm used. In section 3 we will describe the results obtained and the characteristics and advantages of the emergent modular architecture described.

2.1. The robot and the environment

Khepera is 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 by 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 through any angle from vertical to horizontal 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, and the remaining two on the back), and an optical barrier sensor on the gripper capable of detecting the presence of an object within the gripper (the infra-red sensors on the back side of the robot and other available sensors were not used in the experiments described in this paper).

A Motorola 68331 controller with 256 Kbytes of RAM and 512 Kbytes ROM handles all the input-output routines and can communicate via a serial port with a host computer. Khepera was attached to the host computer 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 provides electrical power without using time-consuming homing algorithms or large heavy-duty batteries.

The environment was a rectangular arena 60x35 cm surrounded by walls containing 5 target objects. The walls were 3 cm in height, made of wood, and covered with white paper. Target objects consisted of cylinders with a diameter of 2.3 cm and a height of 3 cm. They were made of cardboard and covered with white paper. Targets were positioned randomly inside the arena.

2.2 The architecture of the controller

Like the majority of people who use evolutionary methods to obtain control systems for autonomous robots (Mataric and Cliff, in press), we decided to implement the controller using a neural network. This decision was based on several reasons: (a) neural networks are resistant to noise, which is massively present in robot/environment interactions and are potentially able of generalizing their behavior to new situations; (b) it is important that the primitives manipulated by the evolutionary process should be at the lowest possible level in order to avoid undesirable choices being made by the human designer (Cliff, Harvey, and Husband, 1993), and synaptic weights and neurons are sufficiently low level primitives; (c) 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 Littman, 1991; Nolfi, Elman and Parisi, 1994; Floreano and Mondada, 1996).

In order to assess the role of modularity, and of emergent modularity in particular, we tried several different network architectures. All architectures had 7 sensory neurons and 4 motor neurons although they differed in their internal organization. The first 6 sensory neurons were used to encode the activation level of the corresponding 6 frontal sensors of Khepera and the seventh sensory neuron was used to encode the barrier light sensor on the gripper. On the motor side the four neurons respectively coded for the speed of the left and right motors and for the triggering of the "object pick-up" and "object release" procedures.

The activation values of the infrared sensors (which can have 1024 different values ranging from 0 to 1023) and of the activation of the light-barrier sensor (which can have two values: 0 or 1023) were encoded in sensory neurons as floating point values between 0.0 and 1.0. The logistic function was used to determine the activation of the motor neurons. The activation of the first two motor neurons controlling the left and right wheels was transformed into 21 different integer values ranging from -10 to +10 (max. speed backward and forward, respectively). The activation of the third and fourth motor neurons controlling the picking-up and releasing procedures, respectively, were thresholded into two values (1 = trigger the corresponding procedure, 0 = do not trigger the corresponding procedure).

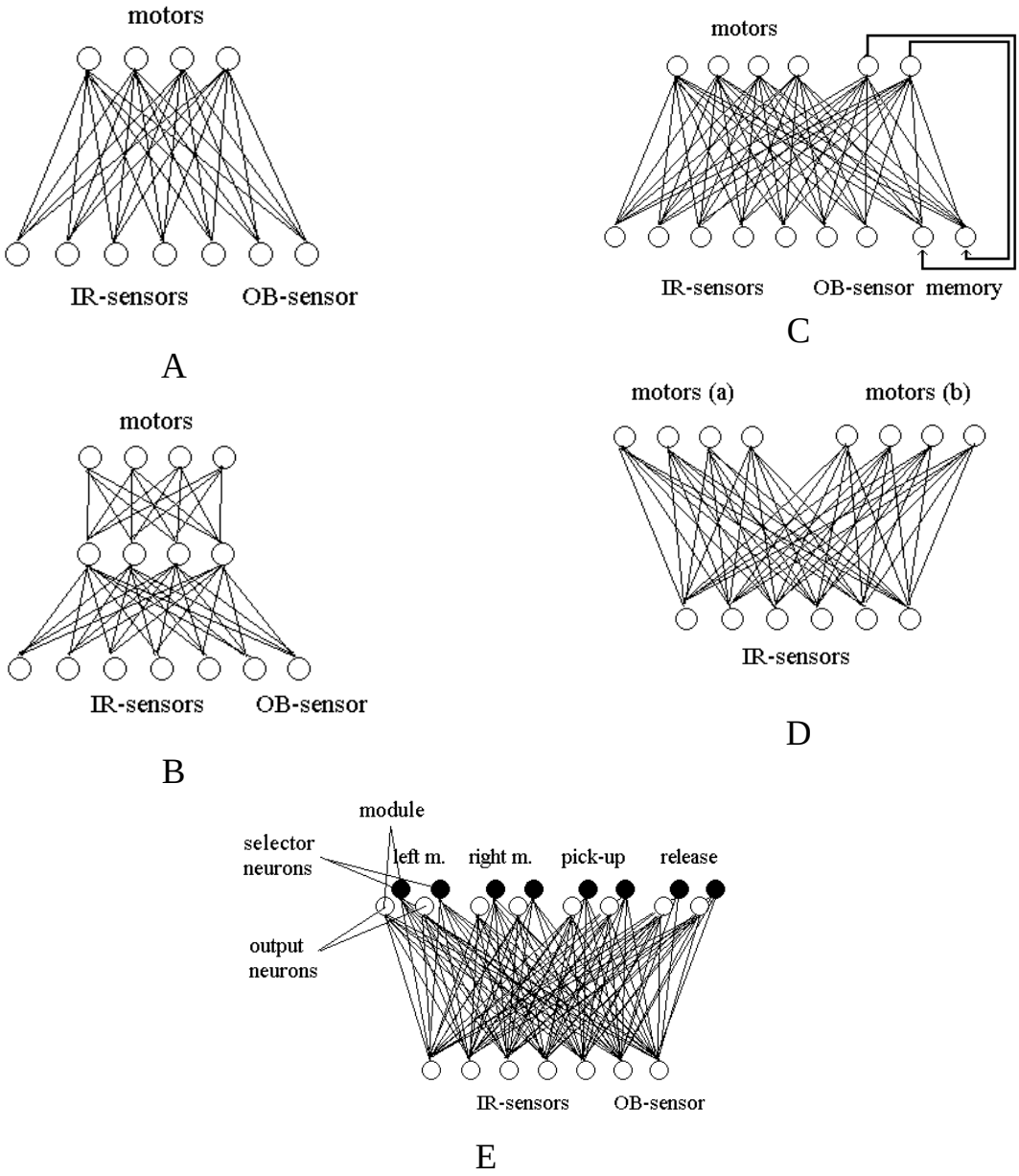

Figure 1. The 5 different architectures used to evolve the controller: (a) a standard feedforward architecture; (b) an architecture with an internal layer of hidden units; (c) a recurrent architecture; (d) a modular architecture with two pre-designed modules; (e) an emergent modular architecture.

The simplest architecture used was a 2-layer feedforward neural network (see Figure 1a). The second architecture was also a feedforward neural network but had an internal layer of four units (Figure 1b). We then tried a recurrent architecture in which the activation level of two additional output units was copied back into two additional input units (Figure 1c). We chose this architecture because it allows the network: (a) to determine which type of information to keep in memory, and (b) to compress into a single pattern of activation information coming from an unspecified number of previous sensory stimuli (for a similar architecture see Elman, 1990). Finally we tried two modular neural architectures (i.e. networks in which different parts or modules had control in different sensory-environmental situations). The first modular architecture (Figure 1d) had two modules, of which the corresponding expected behavior was pre-determined by the designer. The first module (i.e. the sub-network on the left) had control when the robot gripper was empty, and was therefore dedicated to the ability to find a target, while avoiding walls,

recognize it, and pick it up correctly. The second module (i.e. the sub-network on the right) was in control when the gripper was carrying a target and was therefore dedicated to the ability to find a wall while avoiding other targets, stop in front of it and release the target. The partition of the required behavior into these two basic behaviors and into the corresponding neural modules was of course arbitrary, although it seemed to be the most reasonable one given that the robot was expected to perform two very different behaviors depending on the state of the gripper.

The second modular architecture (see Figure 1e) was denoted as an “emergent modular architecture” because it allows the required behavior to be broken down into sub-components corresponding to different neural modules, although it does not require the designer to do such a partition in advance. The number of available neural modules (in this case two for each motor output), the architecture of each module, and the mechanisms that determine their interaction is pre-designed and fixed. However the number of modules actually used by an individual, the combination of modules used each time step, and the weights of the modules themselves are learned during the training phase and are emergent. In particular, the sub-division of the behavior into basic behavior corresponding to different neural modules is emergent. This can be accomplished because the neural structures responsible for the basic behaviors and for the selection mechanisms are represented homogeneously.

This architecture had 16 output units, which, at every time step, gives 4 output values controlling the 4 previously described effectors. Four pairs of output neurons (represented by empty circles) coded for the speed of the left and right motors and for the triggering of the "object pick-up" and "object release" procedures, respectively, and four pairs of selector neurons (represented by full circles) determined which of the two competing output neurons had control over the corresponding robot's effector each time step (the competitor with the corresponding highly activated selector neuron gained control). Each module was composed of two output neurons, two corresponding biases, and 14 connections from sensory neurons. The first output neuron determined the motor output when the module has control, the second output neuron (selector) competes with the selector neuron of the other corresponding module to determine which of the two modules has to take control.

The activation of the sensors and the state of the motors were encoded every 100 milliseconds. However, when the activation level of the "object pick-up" or of the "object release" neurons reached a given threshold, a sequence of action occurred that possibly required one or two seconds to complete (e.g. move a little further 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).

It is important to note that the task chosen is particularly well suited to study the role of modularity because, as described above, the required behavior can be broken down into several basic behaviors that may be implemented in different neural modules. Moreover, the task requires a controller able to produce very different motor responses for similar sensory states. Let us take the case of the robot in front of a target, it should avoid or approach it according to the presence or absence of a target on the gripper (in the two cases the only difference is the state of 1 sensor out of 7). Or else, let us take the case of a robot in front of an object with an empty gripper, it should avoid or approach the object according the type of the object; wall or target (in the two cases the infrared sensors have only slightly different activation values). Our hypothesis is that a modular neural network, that can use different neural modules in different environmental situations, might have an advantage in learning to produce very different motor responses for very similar sensory patterns with respect to a single, uniformly connected, neural network.

2.3. The Genetic Algorithm

To evolve neural controllers able to perform the task described above we used a form of genetic algorithm (Holland, 1975). For each network architecture, we began with 100 randomly generated genotypes each representing a network with the corresponding architecture and a different set of randomly assigned connection weights. This is

Generation 0 (G0). G0 networks are allowed to "live" for 15 epochs, with each epoch consisting of 200 actions (about 8 seconds in the simulated environment using an IBM RISC/6000 and about 300 seconds in the real environment). At the beginning of each epoch the robot and the target objects were randomly positioned in the arena. Epochs terminated after 200 actions or after the first object had been correctly released. At the end of their life, individual robots were allowed to reproduce. However, only the 20 individuals which had accumulated the most fitness in the course of their life reproduced (agamically) by generating 5 copies of their neural networks. These 20x5=100 new robots constituted the next generation (G1). Mutations were introduced in the copying process, resulting in possible changes of the connection weights. Mutations were obtained by substituting 2% of randomly selected bits with a new randomly selected value (as a consequence, about 1% of the bits were actually changed). We tried lower and higher mutation rates in our experiment. This was selected because it gave the best results overall. The process was repeated for 1000 generations.

We also ran a set of simulations in which we used both mutation and crossover. In this case a random single point crossover was performed with a given probability (0.1, 0.5, 1.0). Reproducing individuals were obtained by crossing over one of the 20 individuals with the highest fitness score and one randomly selected individual. However, we did not obtain better performance with respect to simulations without crossover (for this reason we shall present the result obtained using only mutations). This may be due to the fact that the crossover points were randomly chosen. Restricting the crossover points so as to preserve the organization of the network may produce better results (Montana, and Davis, 1989). This may be particularly true for architectures D and E which can be divided into neural modules.

Gene duplication and elimination could also be considered. These operators can be particularly effective in the case of architecture E in order to allow evolution to select the best number of competing neural modules for each motor output. Modularity can be realized at different levels: (a) the genetic level; (b) the nervous system level; (c) the behavioral level. In this paper we will concentrate on (b) and (c). We plan to investigate (a) in the near future.

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 according to which the genetic algorithm operates). One-to-one mapping is the simplest encoding scheme where one and only one 'gene' corresponds to each phenotypical character. In our case, to each connection weight and bias corresponded to a sequence of 8 bits for the genotype which had a total length of: (A) 256, (B) 416, (C) 480, (D) 480, and (E) 1024 bits in the 5 different architectures described. (For more complex encoding schemes also allowing evolution of the neural architecture, see Cliff, Harvey and Husband, 1993; Nolfi, Miglino, and Parisi, 1994; Grau, 1995).

Individual networks were scored by counting the number of objects correctly released outside the arena. However, in order to facilitate the emergence of the ability to achieve the task, individuals were also scored (even if with a much lower reward) for their ability to pick up targets. In addition, it was found important to expose the robots to useful training experiences (i.e. to artificially increase the number of times when the robot, while carrying a target object, encountered another target) in order to force the evolutionary process to select individuals able to avoid targets when the gripper was full. This was accomplished by artificially positioning a new target object in the frontal area of the robot each time it picked up a target during evolutionary training (see also Nolfi, in press).

Without this manipulation of the learning experiences, evolved individuals were unable to avoid targets while carrying an object. We first tried to introduce a penalty term in the fitness function for individuals unable to avoid targets when their gripper was full. However alteration of the learning experiences proved much more effective. The real problem, in fact, is that such cases rarely occurred during training and as a consequence, without altering the learning experiences, there was too little evolutionary pressure to select individuals able to perform the right behavior.

3. Results

We ran 10 simulations for each of the 5 different architectures described above. Each simulation started with populations of 100 networks with randomly assigned connection weights and lasted 1000 generations (about 10 hours using a standard IBM RISC/6000). In the following section we will compare the results obtained for individuals with different architectures, and later we will analyze how modularity is used in emergent modular architectures.

3.1. Results obtained with different architectures

If we measure the average number of epochs (out of 15) in which individuals correctly pick up and then release a target outside the arena for simulations with different architectures we can see how in all conditions an ability to accomplish the correct sequence of behaviors evolves (see Figure 2). Note that epochs terminated after 200 actions or after the first object had been correctly released. As a consequence the max. number of targets that can be released outside the arena is equal to the number of epochs. However, evolved individuals with different architectures vary in the performance achieved at the end of the evolutionary training and in the time needed to reach plateau level performance. If we look at performance of generation 999 we can see how all types of architectures have reached high performances with the exception of the simple feed-forward architecture (A). If we look at performance throughout generations we can see how the emergent modular architectures, after few generations, start to outperform all the other architectures maintaining a difference until generation 500 (this result is even more meaningful if one consider that architecture (E), by requiring a longer genotype with respect to the other architectures, also implies that the genetic algorithm has to search a larger space). A oneway analysis of variance of performance in the five different conditions was performed each 100 generations. The results show that performance in condition (E) is significantly higher than in the other four conditions at generation 199 and performance in condition (A) is significantly lower than the other four conditions at generation 999 ($p < 0.05$).

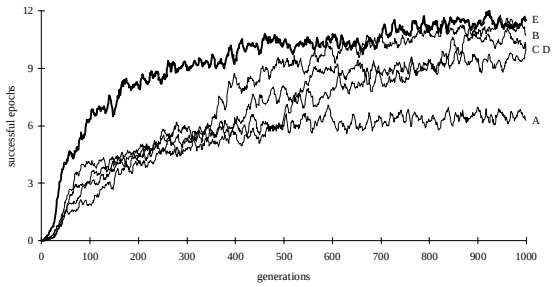

Figure 2. Number of epochs (out of 15) in which individuals with different architectures correctly picked up and then released a target object outside the arena through out generations. Each curve represents the average of the best individuals in 10 different simulations. Data smoothed by calculating rolling averages over preceding and succeeding 3 generations.

By downloading the best controllers of generation 999 (for 10 replications of the simulation) into the robot and testing them in the real environment for 5000 cycles for their ability to clean up the arena by removing 5 randomly placed target objects, we can see that architecture (E) clearly outperforms all other architectures (see Figure 3). The best individuals of 7 (out of 10) with the emergent modular architecture were capable of cleaning the arena without displaying any incorrect behavior while only 1 or 2 individuals (out of 10) with other architectures were capable of accomplishing the task.

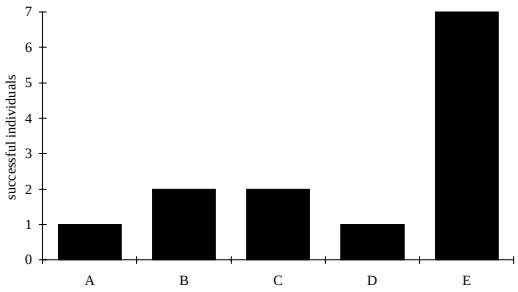

Figure 3. Number of evolved individuals for each control architectures capable of correctly picking up and then releasing outside the arena the 5 targets objects within 5000 cycles without displaying any incorrect behavior (e.g. crashing into walls, trying to grasp a wall, or trying to release a target over another target). A,B,C,D,E

indicate the five different architectures described in Figure 1.

These results show that the emergent modular architecture (E) enables the evolutionary process to find a correct solution to the task earlier than other architectures and in particular earlier than the hand-crafted modular architecture (D). Moreover results show how the emergent modular architecture allows the evolutionary process to select more robust solutions to the task, i.e. controllers which showed only a limited loss of performance when transferred into the real robot.

3.2. How emergent modular neural networks work

The first thing we want to know about our evolved individuals with the emergent modular architecture is: can we find a correspondence between distal description of behaviors and modules? In other words do we find that the evolved individuals use different modules in different environmental situations (e.g. when they have to pick up a target, when they have to release a target, when they have to disambiguate a sensory pattern, when they have to avoid a target etc.)?

The answer to this question is no.

Figure 4 represents the behavior of a typical evolved individual. As we said in the previous section, individuals with emergent modular architecture have two different modules for each of the four motor outputs and therefore can use up to 16 different combination of neural modules. However, the evolved individual described in Figure 4, i.e. one of the most successful, uses only a single module to control the left motor, the pick-up procedures, and the release procedure (LM, PU, and RL) and it uses both neural modules only for the right motor (RM)**.** For an analysis of other individuals see below. What is interesting to note is that those two modules competing for the control of the right motor are both used in all the phases that can be described as distal sub-behaviors: when the gripper is empty and the robot has to look for a target (i.e. when sensor LB is off); when the gripper is carrying a target and the robot has to look for a wall (i.e. when sensor LB is on); when the robot perceives something and has to disambiguate between walls and targets (i.e. when the W/T graph shows the upper or bottom line); when the robot does not perceive anything (i.e. when the 'W/T' graph does not show any line); when the robot is approaching a target (i.e. when sensor LB is off and the perceived object is a target); when the robot is approaching a wall (i.e. when sensor LB is on and the perceived object is a wall); when the robot is avoiding a target (i.e. when sensor LB is on and the perceived object is a target); when the robot is avoiding a wall (i.e. when the sensor LB is off and the perceived object is a wall).

Similar results can be obtained by analyzing the other evolved individuals. When many alternative neural modules are involved it becomes difficult to understand what is going on. However, the general picture remains the same: neural modules or a combination of neural modules does not appear to be responsible for single distal sub-behaviors. On the contrary each sub-behavior is the result of the contribution of different neural modules.

In order to understand the function of modules in these evolved individuals we should abandon the analysis of distal behaviors and concentrate on proximal behaviors (i.e. the way in which individuals respond to different sensory stimuli). In the case of our robot/environment framework, the number of stimuli (intended as the combination of all possible sensor states) to which individuals may be exposed is infinite. However, they can be reduced to a finite number by classifying similar stimuli together.

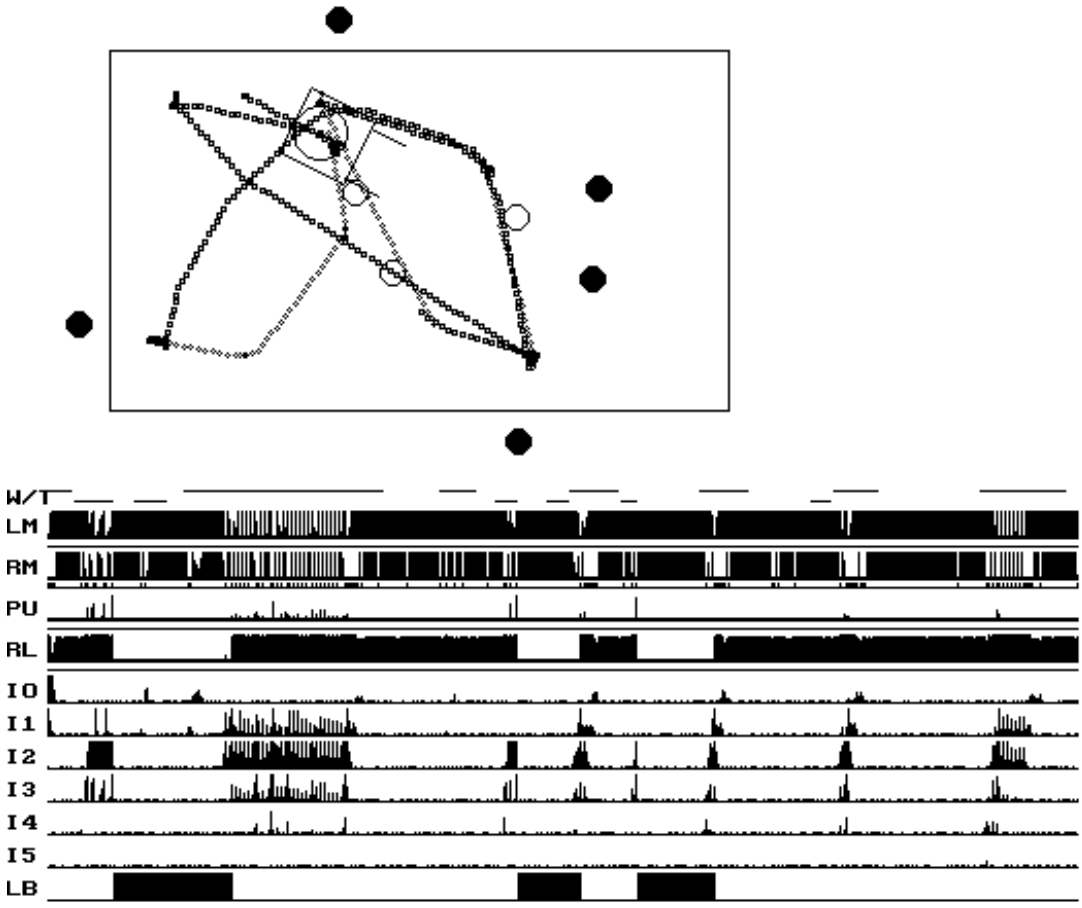

Figure 4. The top part of the figure represents the behavior of a typical evolved individual in its environment. Lines represent walls, empty and full circles represent the original and the final position of the target objects respectively, the trace on the terrain represents the trajectory of the robot. The bottom part of the figure represents the type of object currently perceived, the state of the motor, and the state of the sensors throughout time for 500 cycles respectively. The 'W/T' graph shows whether the robot is currently perceiving a wall (top line), a target (bottom line), or nothing (no line). The 'LM', 'RM,' 'PU', and 'RL' graphs show the state of the motors (left and right motors, pick-up and release procedures, respectively). For each motor, in the top part of the graph the activation state is indicated (after the arbitration between component modules has been performed by the selector neurons) and in the bottom part which of the two competing neural modules has control is indicated (the thickness of the line at the bottom indicates whether the first or the second module has control: thin line segment = module 1; thick line segment = module 2). The graphs 'I0' to 'I5' show the state of the 6 infrared sensors. Finally, the 'LB' graph shows the state of the light-barrier sensor. The activation state of sensor and motor neurons is represented by the height with respect to the baseline (in the case of motor neurons the activation state of the output neurons of the module that currently have the control is shown).

If we assume that stimuli perceived by the robot do not change significantly when the robot modifies its position with respect to a perceived object by turning left or right less than 2 degrees or by moving forth or back less the 2 millimeters (i.e. the same grain used to build the simulator through world samples (see Miglino, Lund, and Nolfi, 1995)) we can reduce the infinite number of different input stimuli to a manageable number. By using a threshold of 2 degrees and 2 millimeters and considering that infra-red sensors are unable to detect objects at a distance of over 40 millimeters, we obtain 180x20=3600 different stimuli for different relative positions of the robot with respect to an object (i.e. 180 different orientations from 0 to 359 degrees multiplied by 20 different distances from 0 to 40 millimeters). In addition, because stimuli also vary according to the type of object

perceived (wall or target) and the state of the light-barrier sensor (on or off), we have a total of 3600x4=14400 possible different sensory stimuli. Figure 5 shows the proximal description of the behavior of the same evolved individual represented in Figure 4. For each of the 3600x4 different environmental stimulation, in addition to the state of the four effectors, the corresponding combination of neural modules that obtain the control is indicated.

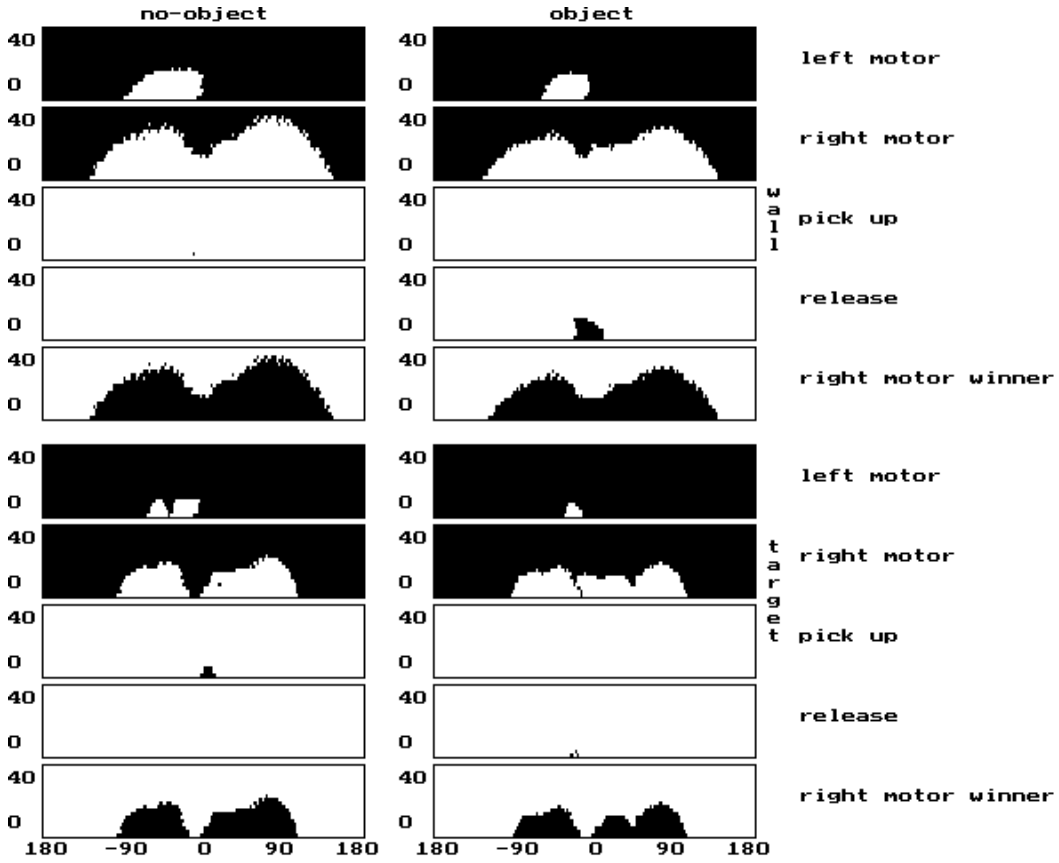

Figure 5. Proximal representation of the behavior of an evolved individual. The rectangular maps labeled 'left motor', 'right motor', 'pick up', and 'release' represent the activation state of the four actuators (after the arbitration between component modules has been performed by the selector neurons) for 180 different orientations over 360° and 20 different distances (from 0 to 40 mm) with respect to a perceived object. For graphic reasons the activation states of the motor neurons are divided into two classes, positive and negative speeds (represented with black and white color respectively), although the left and right motors can take 20 different speeds. The 'right motor winner' maps represent which of the two modules controlling the right motor is in charge for each combination of orientations and distance with respect to a perceived object (the two modules are represented in white and black, respectively). The group of 5 maps is replicated for 4 different conditions: perceived object is a wall and no object is in the gripper (top left maps); perceived object is a target and no object is in the gripper (bottom left maps); perceived object is a wall and an object is in the gripper (top right maps); perceived object is a target and an object is in the gripper (bottom right maps).

If we analyze in which environmental situations the two different combinations of neural modules obtain control we can see how, in the case of the individual represented in Figure 5, the neural module shown in black obtains the control when the perceived object is within a given angle (from about -100° to 100°) and a given distance (about 30mm) with respect to the robot. However, the extension of the "area" in which the module represented in black obtains control varies significantly according to whether the perceived object is a wall or a target (it is much larger in the case of a wall) and, although much less significantly, whether the robot has an object in the gripper or not (it is larger in the second case).

As can be seen, there is almost a one to one correspondence between the combination

of neural modules that have control (right motor winner) and the speed of the right motor, which is the only motor affected by the alternation of the two competing neural modules in this individual. The speed of the right motor is negative when the black neural module is activated and positive otherwise. Therefore the weights that determine which of the two neural modules has control have the main responsibility in determining the speed of the right motor. However, the weights of the two neural modules are responsible for differentiating the speed of the right motor in a very important environmental situation (when both angle and distance are close to 0) depending on the presence or not of an object in the gripper (see the left and right 'right motor' maps in the bottom of Figure 5).

By observing the close descriptions of behavior of other evolved individuals (obtained by replicating the simulation) it appears that different combinations of modules are used to produce different motor responses for similar sensory stimuli when necessary. This happens most of the time, as in the case of the individual described in Figure 5, when the robot has an object on its frontal side and must decide whether to approach or avoid it or whether to try to pick it up or not. Individuals with the other architecture described appear less able to produce sharp discontinuities in behavior.

The need to produce very different motor responses for very similar sensory stimuli is related to the complexity of the task. This type of architecture may consequently be expected to scale up more easily than other non-modular architectures. We have not jet applied this architecture to more complex tasks than the one described in this paper. However we have observed that while homogeneous architectures are perfectly able to solve simpler tasks like the ability to explore an arena surrounded by walls (Nolfi, Floreano, Miglino, and Mondada, 1994) or the ability to recognize, approach and to remain close to target objects while avoiding walls in an environment identical to that described in this paper (Nolfi, 1996), the emergent modular neural network outperformed other architectures in garbage collecting task. Therefore, one can hypothesize that the advantage of the emergent modular architecture will increase with increasing task complexity.

3.3 Why there is no correspondence between evolved modules and distal description of behaviors.

There are at least two reasons that can explain why there is no correspondence between distal description of behaviors and neural modules in evolved individuals. First, one should consider that different distal behaviors usually require behavioral responses that differ only partially. Think of the following situations: object in the gripper or not. If the robot has an object in the gripper it should avoid targets and approach walls and viceversa when it has the gripper empty it should approach targets and avoid walls. However the perceived object can be correctly classified only from a small number of relative positions (see Nolfi, 1996). Therefore, in all the other cases, there is no reason to have different behaviors involving different neural modules. Conversely having two different neural modules for the cases "object in the gripper or not", as in architecture D described above, requires the additional cost of learning the same behavior in all cases in which there is no reason to have different motor responses. Alternation of different neural modules is required only when the agent is expected to produce different motor responses for similar input patterns, and this happens only in some cases in different distal behaviors.

One second reason is that the division of a requested behavior into a set of basic behavior that correspond to distal description often requires a very complex behavioral selection system. Let us take the basic behaviors: avoiding a target, avoiding a wall, approaching a target, and approaching a wall. All these behaviors are probably easy to implement (or to learn); however they require a selection mechanism that, in order to be able to select the right module at the right time, should always be able to correctly classify walls and target and this is a complex task that can be accomplished only in some environmental circumstances. In our emergent modular architecture, because the neural modules and the selection mechanism are represented homogeneously and evolve at the same time,

solutions in which both the components are kept as simple as possible are selected.

An interesting property of the emergent modular network is that it does not require specification of the number of modules corresponding to basic behaviors into which the desired behavior should be broken down. It is only necessary to specify the max. number of available neural modules for each motor which then determines the max. number of different combinations of neural modules that can be exploited. In the present paper we used an architecture with 8 neural modules (two neural modules for each of the four motor functions) which can produce up to 16 different combinations of neural modules. However, only about 6 neural modules and about 5 combinations of neural modules were used, on average, by evolved individuals. The number of neural modules actually selected by evolution can be expected to be, on average, proportional to the complexity of the task.

This is an important property of the architecture. In fact if we assume that the best way to break down a behavior into basic behaviors corresponding to different neural modules is to take into account the close description of behaviors, and we also assume that there is a complex mapping between close and distal description of behavior we will conclude that: as the designer will be unable to specify the best way to break down the required behavior into basic behaviors, he will also be unable to specify how to select the number of basic behaviors. Nor will the designer able to decide how to combine different neural modules for each time step.

4. Conclusions

We have presented an architecture, known as emergent modular architecture, in which for each output function two or more alternative neural modules compete for control and two or more other corresponding neural modules determine which competitor gains control. This architecture, by using a uniform representation for modules responsible for basic behavior and mechanisms responsible for behavior selection, allows not only the control structures responsible for basic behaviors and behavior selection but also the break down of the target behavior into basic behaviors to be obtained through an adaptation process.

By evolving controllers for mobile robots for the purpose of performing a non trivial task and by comparing the results obtained using this architecture with other neural architectures we showed that the emergent modular architecture outperforms all other architectures and in particular the modular architecture in which the break-down of the required behavior into basic behaviors is hand-crafted.

The analysis of the evolved individuals with the emergent modular architecture showed that modularity and action selection can be useful in tasks, like that presented in this paper, in which very different motor responses should be produced for similar sensory patterns. In other words modularity appears useful in tasks which require complex behavior from the point of view of proximal description. Moreover, the analysis of our results showed that there is no correspondence between evolved modules and distal description of behaviors. In fact, the sequence of sensory-motor loops that can be described as basic behaviors from the observer's point of view are the result of the contribution of different neural modules in evolved individuals with the emergent modular architecture.

The fact that the process of breaking down the required behavior into basic behaviors should take into account the proximal description of behavior itself can impose serious limitations on the engineering-oriented approach to behavior-based robotics. This is because there is a complex mapping between distal and proximal description of behavior and therefore we cannot expect the experimenter, who has direct access only to the distal description of behavior, to have a correct picture of the corresponding proximal descriptions, excepted for trivial cases. Even the use of learning in the development of the modules responsible for the basic behavior and/or of the mechanisms responsible for action selection might suffice. Hand-crafting the process of breaking down the required behavior into basic components leads to constraints that may limit the adaptation process to the borders set by the experimenter.

Our claim that behavioral modules should be allocated by considering the proximal description and not the distal description of behavior, as is usually done, may only be true (or particularly true) for neurocontrollers

where a homogeneous set of weighted sums are often requested to account for sharp discontinuities in behavior. However, similar impressions have been reported by other researchers following different approaches. Mahadevan and Connel, for example, who developed a box-pushing controller for an autonomous robot using a subsumption architecture wrote "Obviously, there may be several ways of decomposing a given task, and coming up with a good decomposition is a nontrivial problem" (Mahadevan, and Connel; 1992, p.363).

Acknowledgment

This research has been granted by the Coordinated Project on real Time Computing in Real World of C.N.R, Italy. The author thanks the anonymous referees for valuable suggestions on the manuscript.

References

Ackley, D. H., & Littman, M. L. (1991). Interactions between learning and evolution, in: C. G. Langton, J. D. Farmer, S. Rasmussen, C. E. Taylor (eds.), *Artificial Life II*, Reading, Mass., Addison-Wesley.

Braitenberg, V. (1984). *Vehicles: experiments in synthetic psychology*, Cambridge, MA: MIT Press.

Brooks, R. A. (1986). A roboust layered control system for a mobile robot, *IEEE Journal of Robotics and Autonomation*, 2, 14-23.

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

Colombetti, M., Dorigo, M., & Borghi G. (1996). Behavior analysis and training. A methodology for behavior engineering. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, (26) 1, 29-41

Dorigo, M., & Schnepf, U. (1993). Genetic-based machine learning and behaviour based robotics: a new synthesis. *IEEE Transaction on Systems, Man, and Cybernetics*, (23) 1, 141-154.

Elman, J. L. (1990) Finding structure in time, *Cognitive Science*, 14, 179-211

Floreano, D., & Mondada, F. (1996). Evolution of plastic neurocontrollers for situated agents, in: P. Maes, M. Mataric, J-A. Meyer, J. Pollack, & S. Wilson. (eds.), *From Animals to Animats IV*, Cambridge, MA: MIT Press.

Gruau, F. (1995). Automatic definition of modular neural networks. *Adaptive Behavior*, 2, 151-183.

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

Maes, P. (1992). Learning behavior networks from experience, in*:* F. J. Varela, P. Bourgine (eds.), *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, Cambridge, Mass, MIT Press/Bradford Books.

Mahadevan, S., & Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55, 311-365.

Mataric, M. J. (1992). Behavior-based control: Main properties and implications. *Proceedings of the IEEE International Conference on Robotics and Autonomation*, Workshop on Architectures for Intelligent Control Systems, Nice, France.

Mataric, M. J., & Cliff, D. (in press). Challenges in evolving controllers for physical robots, in "Evolutionary Robotics", special issue of *Robotics and Autonomous Systems.*

Miglino, O., Lund, H. H., & Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life*, (2) 4, 417-434.

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

Montana, D. J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms, in: *Proceedings of Eleventh Joint Conference on Artificial Intelligence*, Vol 1, Palo Alto, CA: Kaufmann.

Nolfi, S. (1996). Adaptation as a more powerful tool than decomposition and integration, in T. Fogarty and G. Venturini (eds), Proceedings of the workshop on Evolutionary computing and Machine Learning, 13th International Conference on Machine Learning, Bari.

Nolfi, S. (in press). Evolving non-trivial behaviors on real robots: a garbage collecting robot. *Robotics and Autonomous Systems*.

Nolfi, S., Elman, J.L., & Parisi, D. (1994). Learning and Evolution in Neural Networks. *Adaptive Behavior*, 1, 5-28.

Nolfi, S., Floreano, D., Miglino, O., & Mondada, F. (1994). How to evolve autonomous robots: different approaches in evolutionary robotics, in: R.A. Brooks and P. Maes (eds.), *Proceedings of fourth International*

Conference on Artificial Life, Cambridge, Mass, MIT Press.

Nolfi, S., Miglino, O., & Parisi, D. (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., & Parisi, D. (1993). Self-selection of input stimuli for improving performance. In: G. A. Bekey (ed.) *Neural Networks and Robotics*, Kluwer Academic Publisher.

Parisi, D., Cecconi, F., & Nolfi, S. (1990). Econets: Neural networks that learn in an environment. *Network*,1,149-168.

Sharkey, N. E., & Heemskerk, N. H. (in press). The neural mind and the robot, in A. J. Browne (Ed.) *Current Perspective in Neural Computing*, IOP press.

Scheier C., & Pfeifer, R. (1995). Classification as sensory-motor coordination: A case study on autonomous agents, in: F. Moran, A. Moreno, J.J. Merelo, P. Chacon (Eds.) *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life*, Springer Verlag.

Scheier, C., & Lambrinos, D. (1995). Adaptive classification in autonomous agents. *Technical Report*, AILab, Computer Science Department, University of Zurich.