Institute of Psychology
C.N.R. - Rome

Evolving non-trivial behavior on autonomous robots: Adaptation is more powerful than decomposition and integration

Stefano Nolfi
Department of Neural Systems and Artificial Life
Institute of Psychology, National Research Council (C.N.R.)
15, Viale Marx
00137 - Rome - Italy
voice: 0039-6-86090231
fax: 0039-6-824737
e-mail: stefano@kant.irmkant.rm.cnr.it
www:http://kant.irmkant.rm.cnr.it/gral.html

April 1997

Published in: Nolfi, S. (1997). Evolving non-trivial behavior on autonomous robots: Adaptation is more powerful than decomposition and integration. In T. Gomi (Ed.), Evolutionary Robotics, Ontario (Canada): AAI Books, 1997.

Evolving non-trivial behavior on autonomous robots: Adaptation is more powerful than decomposition and integration

Stefano Nolfi
Institute of Psychology, National Research Council
15, Viale Marx - 00187 - Rome - Italy
voice: ++39-6-86090231 fax:++39-6-824737
e-mail: stefano@kant.irmkant.rm.cnr.it
http://kant.irmkant.rm.cnr.it/nolfi.html

1. Introduction

Recently a new way of building control systems, known as behavior based robotics, has been proposed to overcome the difficulties of the traditional AI approach to robotics (Brooks, 1986). This new approach is based upon the idea of using simple sensorimotor processes, operating in parallel, to enable robots to react quickly and robustly in noisy environments.

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", and designing an action selection or coordination mechanism able to ensure that only the correct basic behavior has 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.

We believe that the process of breaking down the required behavior into sub-components should be the result of an adaptation process and not of a decision of the experimenter. The most straightforward way to shape behavior through adaptation is to use the evolutionary robotics approach (Cliff, Harvey, and Husbands, 1993) in which behaviors are developed in close interaction with the environment and in which the human intervention is limited to the specification of a rule for determining how much a given behavior approximates to the one desired.

In this paper we will support this hypothesis by showing how in two different cases a more simple and robust solution can be obtained by letting the entire behavior emerge through an evolutionary technique than by trying to design a set of modules and to coordinate them. In the first case we will analyze the case of a Khepera robot that should be able to classify objects of different shapes by finding and remaining close to the object of type A (cylinders) and by avoiding and ignoring objects of type B (walls). In the second case we will analyze the case of a robot of the same type but with a gripper module that should be able to find and pick-up cylindrical objects and to release them outside an arena surrounded by walls.

2. Methodological issues

The experiments we describe in the next Sections involve the miniature mobile robot Khepera (Figure 1) 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 able to detect the presence of an object in the gripper (the two back infra-red sensors and others 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 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.

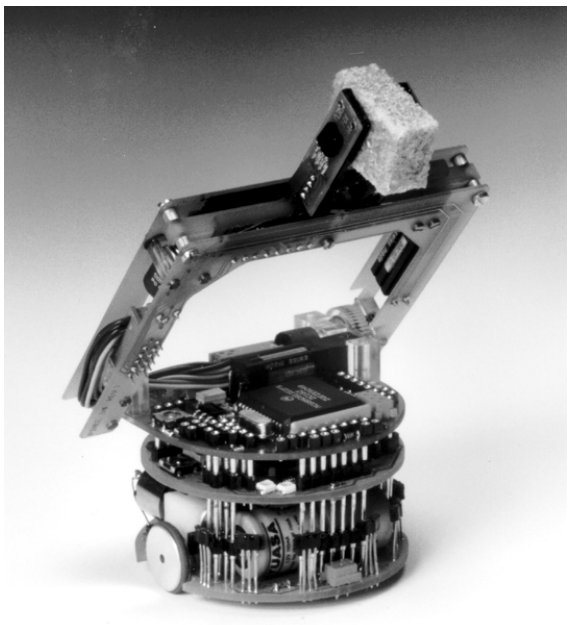

Figure 1. The Khepera robot.

To train the controller for the robot we used a simulator of the robot and of the environment. The resulting controllers where then downloaded and tested on the real robot. This procedure was used to reduce the time needed for the training process. The most complex experiment, that described in Section 4, took about one day for each simulation on the computer and it would have taken about one year on the real robot.

A sampling procedure was used to calculate the activation state of the infra-red sensors. The walls and the target objects were sampled by placing the physical Khepera in front of one of them, and by letting it turn 360^{o}, recording, at the same time, the state of the infra-red sensors at different distances with respect to the 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 matrices of activation were 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 on the relative position of Khepera and of the objects in the simulated environment. (When more than one object was within the range of activation of the sensors, the resulting activation is computed by summing the activation contribution of each object). This sampling procedure may prove to be time consuming in the case of highly 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 throughout 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 each 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 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 and computations were carried out with floating point precision. Motor actions that caused the robot to crash into the walls were not executed in the simulated environment. Therefore, the robot can get stuck at the walls if it was unable to avoid them. In contrast, when the arm crashed into a piece of trash, the trash was moved to a new random location within the environment (for more about methodological issues see Nolfi, Florano, Miglino, and Mondada, 1994; Miglino, Lund, and Nolfi, 1995).

3. Comparing the decomposition/integration and the evolutionary approach

In the first experiment we describe we tried to develop a control system for a Khepera robot that could distinguish between objects with different shapes.

The environment of the robot is a rectangular arena of 60x35 cm surrounded by walls which contains a target object. The walls are 3 cm in height, made of wood, and are covered with white paper. The target object consists of a cylinder with a diameter of 2.3 cm and a height of 3 cm. It is made of cardboard and covered with white paper. The target is positioned in the center of the arena.

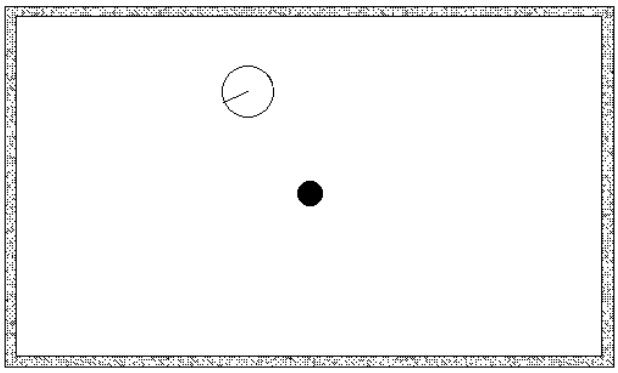

Figure 2. The environment and the robot.

The robot has to distinguish between walls and target objects. In particular we want the robot to explore the environment, avoiding walls and remaining close to targets when it finds them.

In section 3.1 we will describe an attempt to solve the task by using a decomposition and integration approach, while in section 3.2 we will describe a similar attempt using an evolutionary approach. In both cases we tried neural networks with different architectures to implement the controllers.

3.1 Designing by decomposition and integration

In order to accomplish the task described we can break down the required behavior into simpler behaviors and develop a set or modules able to accomplish the required sub-behaviors. In our case we can break down the required behavior into: (1) moving in order to explore the environment, (2) recognizing objects, (3) avoiding objects, (4) approaching and remaining close to an object. Once we have modules able to produce these sub-behaviors we have to design a coordination mechanism that can decide which module has to take control each time step or else design the modules and the corresponding behaviors in such a way that, by letting all of them run in parallel, the interference arising from the interaction between different modules does not impair the accomplishment of the task.

The first module, the third and the fourth modules (i.e. exploring the environment, avoiding objects, and approaching objects) are easy to design and have been implemented several time on Khepera and on other robots. On the contrary, the second module (i.e. classifying sensory stimuli) is not so easy to design. However, because fortunately we know if an input pattern corresponds to a wall or to a target, we can use a supervised learning procedure to train a system (for example a neural network) to classify the two types of input stimuli.

We tried 3 different architectures: (a) a feedforward architecture with two layers, an input layer with 6 neurons (coding the activation of the 6 frontal infrared sensors of Khepera) and one output neuron (coding with 0 for wall and 1 for targets); (b) an architecture with an additional internal layer with four units; (c) an architecture with an additional internal layer with 8 units. For each architecture we ran 10 training processes starting with different randomly assigned initial weights. Networks were trained by back-propagation (Rumelhart, Hinton, and Williams 1986) for 5000 epochs. A learning rate of 0.02 and no momentum were used. During each epoch, networks were exposed to the sensory patterns perceived by the robot at 20 different distances and at 180 different angles with respect to a wall and to a target, for a total of 7200 different patterns.

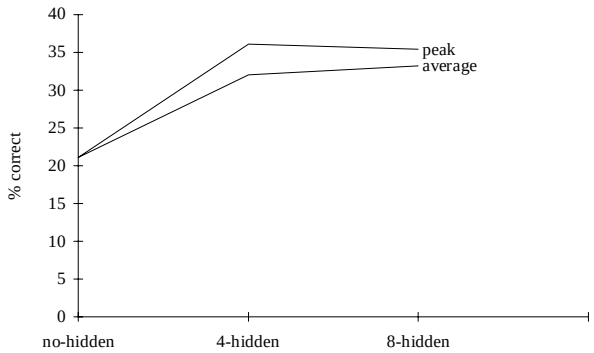

Figure 3. Percentage of positions from which sensory stimuli are correctly classified by network with no hidden units, 4 hidden units, and 8 hidden units. Average and peak performance of 10 different simulations for each condition. For 20 different distances and 180 different angles (3600 different positions in all) networks were required to produce an output above or below the threshold, depending on the type of object. Responses were considered correct if, in a given position, the network was able to correctly classify both the wall and the target.

Figure 3 shows the percentage of positions from which the networks were correctly able to classify the two types of stimuli (i.e. to produce an activation value below 0.5 in the case of a wall and above 0.5 in the case of a target) for each of the three different architectures. As can be seen, networks were able to correctly classify only 22% of the cases for networks without hidden units and about 35% of the cases for networks with the additional layer of 4 hidden units. The addition of other hidden units did not allows to obtain better performance.

The fact that only some of the stimuli can be correctly classified can be explained if we consider that, given the sensory apparatus of the robot, objects can be disambiguated only at a given angle and distance. In other words they are ambiguous in the other cases. If we look at Figure 4, which represents the positions (i.e. the combination of angle and distance) from which networks are able to correctly classify the sensory patterns, we see that stimuli can be correctly classified if the objects are not more than 120^{o} to the left or the right of the robot face and no more than 32mm away. In addition, there are two zones in which the objects cannot be correctly disambiguated even though they correspond to positions located at about 60^{o} to the left or the right and a short distance away from the objects.

It is also interesting to note that the zones in which stimuli can be correctly disambiguated are not symmetrical. This has to do with the fact that different sensors, even if identical from the electronic point of view, actually respond differently. As a consequence, it is clear that whether stimuli are ambiguous or not is a function of both the structure of the stimuli themselves and of the sensory apparatus of the agent.

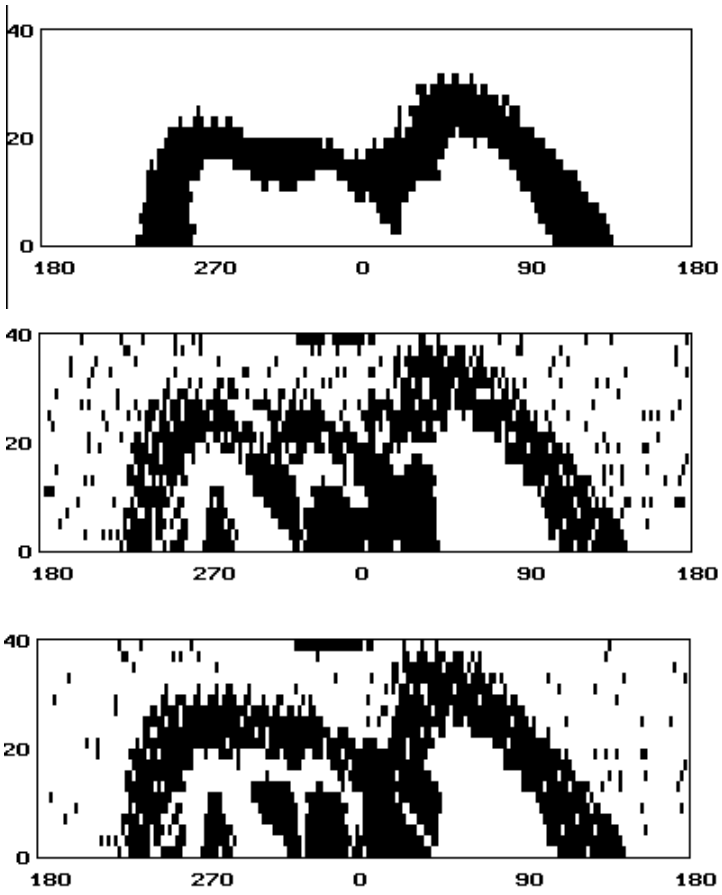

Figure 4. Stimuli correctly classified for each combination of distance and angle of the robot with respect to the objects. The three pictures (from top to bottom) represent the result for the best simulation with no hidden, 4 hidden, and 8 hidden units, respectively.

Given these results we can see that a simple feedforward neural networks, irrespective of the number of internal neurons, is unable to correctly classify the incoming input patterns in our environment. In fact, most of the times the network produces incorrect classifications as shown in Figure 2 and 3.

A possible solution to this problem is to add an additional module capable of deciding when the incoming stimuli can be correctly classified by a network of the type described above. When stimuli cannot be classified, the robot can continue to approach the object until a classifiable stimulus is encountered.

Scheier and Pfeifer (1995), who developed the control systems for a Khepera robot that performed a task very similar to the one described in this paper, proposed and implemented another interesting solution (see also Scheier and Lambrinos, 1996). Their environment is an arena surrounded by walls containing large and small pegs and a home base with a light source attached to it. The robot has to bring the small pegs to its home base and as a consequence has to distinguish between small pegs from large pegs and walls. In order to accomplish this task the authors designed a set of elementary behaviors (move forward, turn toward objects, avoid obstacles, grasp, and bring to the nest) and allowed them all to run in parallel in order to obtain the desired behavior. Because the robot was programmed to turn around objects and the shape of the object determined the angular velocity of the turning, they decided to train a network to associate the vectors of angular velocities that corresponded to small pegs with grasping behavior and the other vectors of angular velocity with avoiding behavior.

It thus appears possible to find a solution to the problem of classification in this robot-environment context. However, in the next section we will show that by using adaptation instead of decomposition-and-integration a simple feedforward network without hidden units can be trained not only to correctly classify the two types of objects but also to perform the entire required behavior without the need to add any other neural structure.

3.2 Designing by evolution

To evolve neural controllers able to perform the task described in section 3 we began with 100 randomly generated genotypes, each representing a network with a different set of randomly assigned connection weights. This was Generation 0 (G0). G0 networks were allowed to "live" for 5 epochs, with each epoch consisting of 500 actions. At the beginning of each epoch the robot was randomly positioned in the arena far from the target. At the end of their life, individual robots were allowed to reproduce. However, only the 20 individuals which had accumulated the greatest fitness (see below) 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). Random mutations were introduced in the copying process, resulting in possible changes of the connection weights. The process was repeated for 100 generations.

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).

Individuals were scored for the number of cycles spent at a distance lower that 8 cm from the target. This meant that individuals had a probability to reproduce that was proportional to their ability to find the target quickly and to remain close to it. The activation of the sensors and the state of the motors were encoded every 100 milliseconds.

We tried 3 different architectures and for each of them we ran 10 simulations starting with different randomly assigned weights. All architectures had 6 sensory neurons which encode the activation level of the corresponding 6 frontal sensors of Khepera and 2 motor neurons which encode the speed of the left and right motors of Khepera. The three architectures differed in the number of internal units: the first architecture did not have any internal layer, the second had an internal layer with 4 units, and the third had an internal layer with 8 units.

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 respectively: 112, 304, and 592 in the 3 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).

If we look at the fitness of individuals throughout generations we can see how, after a few generations, the best individuals, by incorporating an ability to avoid walls, explore the environment and keep close to the target, are able to earn very high fitness rate by spending most of their time close to the target objects (see the graph on the left side of Figure 5).

It should be noted that networks without internal units are able to solve the task perfectly well. As a consequence the addition of internal units does not produce any improvement in performance and, indeed, result in less efficient individuals in average (see the graph on the right side of Figure 5). This can be explained by the fact that the addition of useless internal neurons, which require longer genotypes, merely enlarge the space to be searched by the evolutionary process.

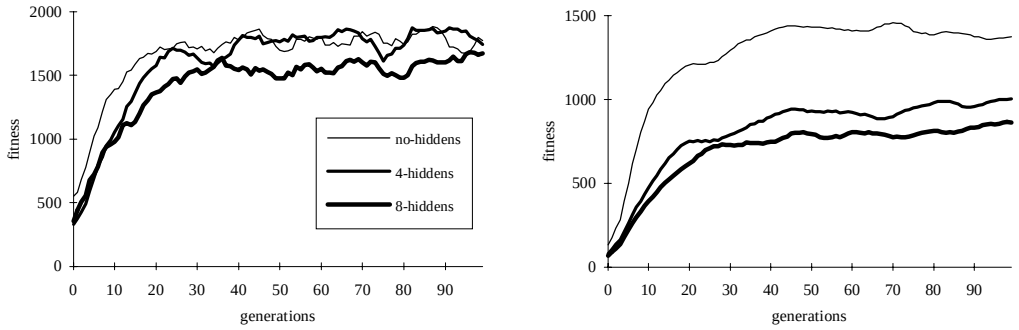

Figure 5. Fitness (i.e. number of cycles, out of 2500, spent close to the target) throughout generations for the three different architectures. The graph on the left shows the performance of the best individual in each generation, while the graph on the right shows the average performance of each generation. Each curve represents the average result of 10 simulations starting with different initial weights.

Nevertheless individual networks have been trained in simulation, the evolved individuals were then downloaded and tested on the real robot and were find to be capable of performing the task perfectly well. No significant difference was observed between behavior in the simulated and real environments in most of the evolved individuals.

Figure 6 shows the behavior of a typical evolved individual. The robot, after being placed in front of a wall on the right hand side of the environment, recognizes and avoids the wall, recognizes and avoids the new wall it encounters, and finally finds and keeps close to

the target. All individuals, like the one shown in the figure, never stop in front of the target, but start to move back and forth as well as slightly to the left and right remaining at a given angle and distance with respect to the target (see graphs ‘a’ and ‘d’ representing the angle and the distance of the robot with respect to the closest object over time).

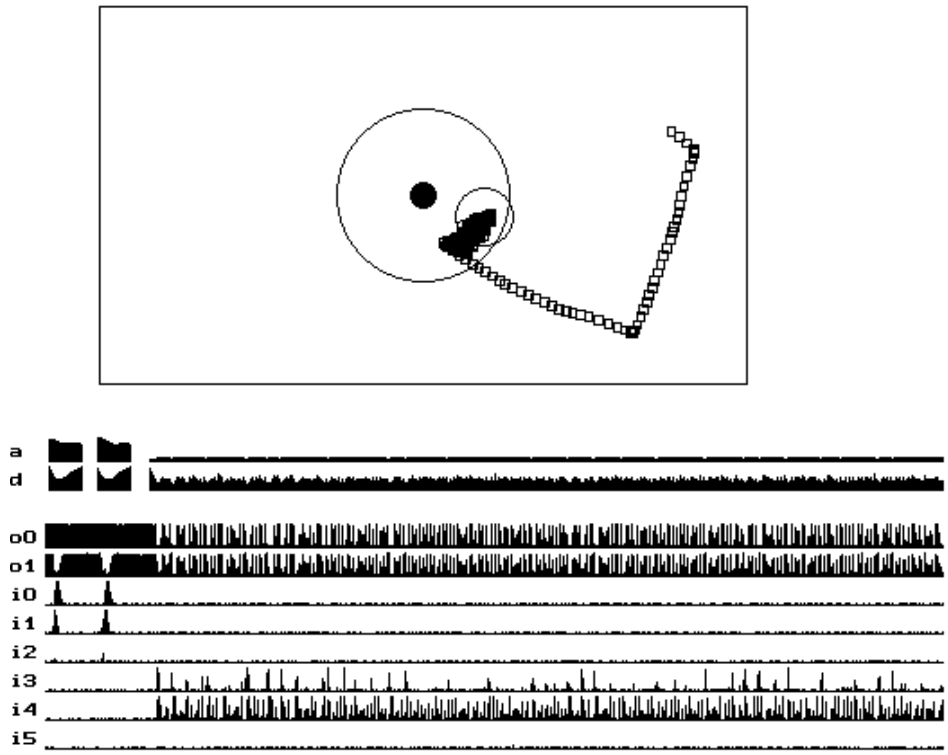

Figure 6. The top part of the figure represents the behavior of a typical evolved individual without internal units. The lines represent walls, the full circle in the center of the arena represents the target object, the large empty circle around the target represents the area in which the robot is rewarded, the small empty circle represents the position of the robot after 500 cycles, finally the trace on the terrain represents the trajectory of the robot. The graphs in the bottom part of the figure represent the angle and the distance of the robot with respect to the closest object (a, d), the state of the two motor neurons (o0, o1), and the state of the 6 infrared sensors (i0 to i5) over time.

3.3 Decomposition-and-integration and evolution lead to qualitatively different solutions

Interestingly enough, the positions relative to the target that individual robots (with and without internal units) maintain once the target has been found, are not located in the discriminative area, that is in the area in which targets and walls can be correctly classified by a network without hidden units (see Figure 3). On the contrary, individual networks do usually start to move back and forth when they reach an area that overlaps the border between the discriminative and non discriminative areas as is shown in Figure 7.

By analyzing the strategy of the other evolved individuals we observed that, because of the different initial conditions of the evolutionary process, these individuals varied in the relative positions assumed in front of the target (some stopped with the target in front or almost in front of them, others with the target on the right side, and only a few with the target on the left side). However, all evolved individuals remained close to the target, moving back and forth in front of it. In addition, in most cases, evolved individuals remained close to the target by assuming relative positions which were not located in the discriminative area (i.e.

from which the two type of objects could not be correctly classified by a network trained with back-propagation).

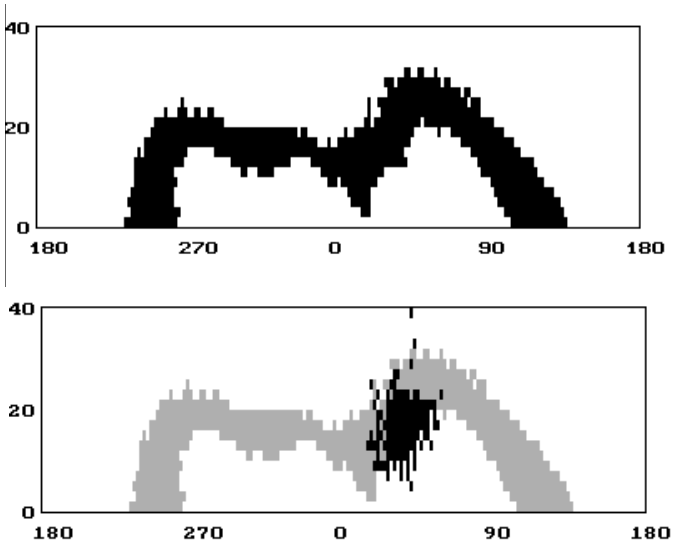

Figure 7. The top picture represents the stimuli that can be correctly classified by an evolved network without internal units (same data as Figure 4). The picture on the bottom represents, in black, the relative positions (angular and distance combinations) that a typical evolved individual assumes with respect to a target once the target has been reached. Each point represents the combination of angle (from -180 to +180 degrees) and distance (from 0 to 40 mm) with respect to the target during one cycle.

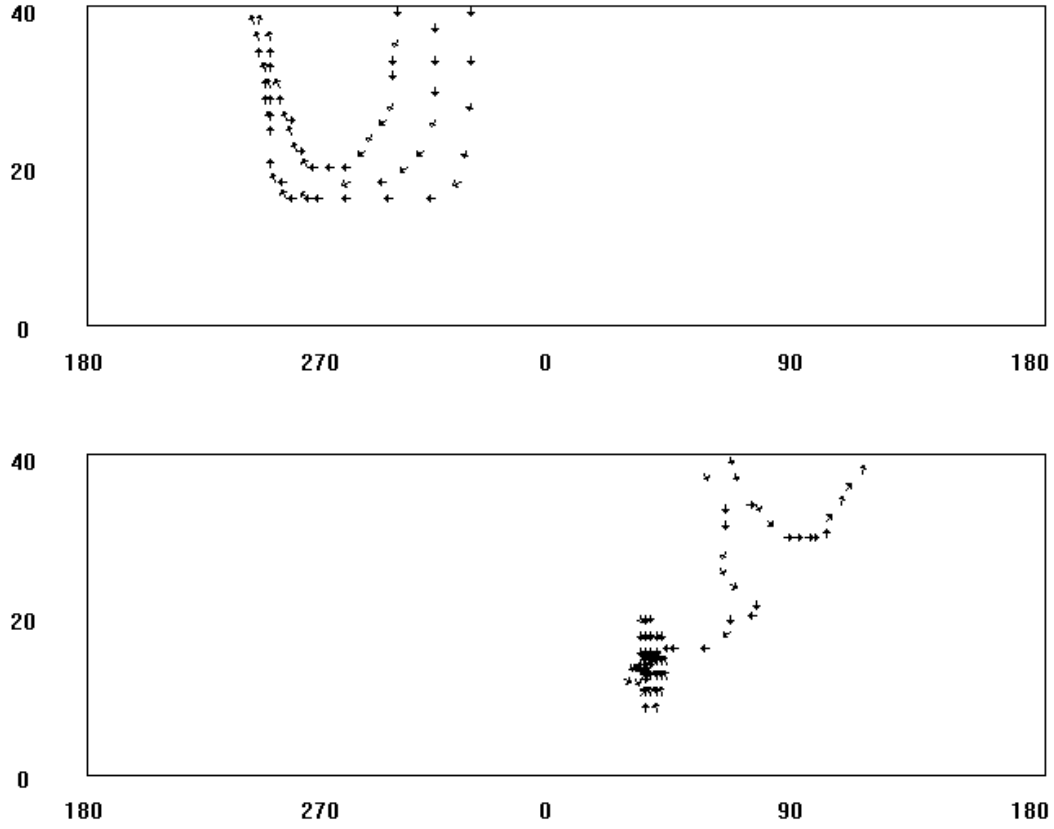

Figure 8. Angular trajectories of the same individual described in Figure 6 and 7 close to a wall (top graph) and to a target (bottom graph). The picture was obtained by placing the individual in a random position in the

environment, leaving it free to interact with the environment for 500 cycles, and recording the change in the relative positions with respect to the two objects for distances lower than 40 mm. For the sake of clarity arrows are used to indicate the relative direction but not the amplitude of movements.

The evolved behavior can be described as a dynamical system and the relative position in which individuals start to move back and forth while remaining in about the same relative position with respect to the target can be described as an attractor since the robot's trajectory converges on the same relative positions regardless of the direction of approach to the target. This is more clearly shown in Figure 8 and 9 which shows the trajectory of the movements that the same individual produced when it come close to a wall or the target. As can be seen, when the individual reaches a distance of about 20mm from an object it avoids walls while it continue to approaches targets until it reaches the attractor area located at a distance of about 15 mm and an angle of about 45 degrees. As can be seen, the trajectory of the motor responses in this area all converge to the center of the area itself allowing the individual to keep more or less the same relative position.

It is interesting to note that, as shown in Figure 8, the individual was able to reach the attractor area only if it approached the target from the right side (i.e. at an angle of between about 30 and 60 degrees). When it approached targets from the left side (i.e. the side from which it approached walls 70% of the times), it erroneously avoided the target.

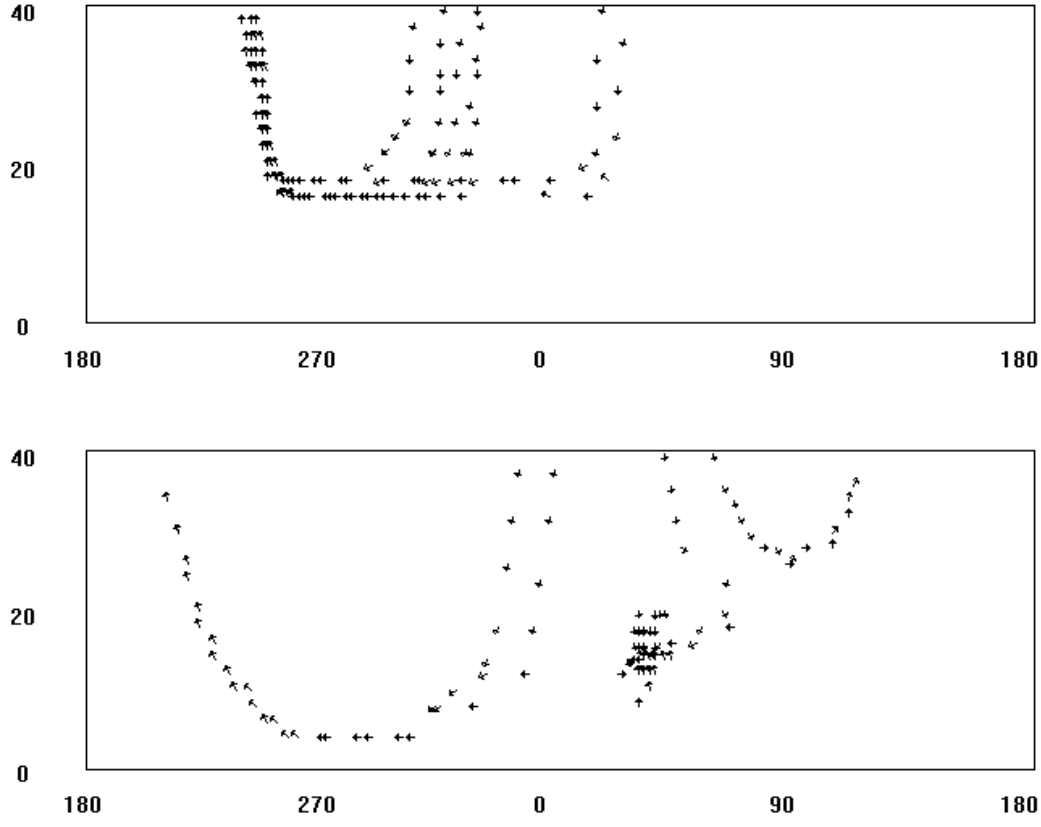

Figure 9. Angular trajectories of the same individual described in Figure 8 starting from a different random selected initial position.

It should be clear at this point that the type of solution found by the evolutionary process is qualitatively different from those that we and other researchers (e.g. Scheier and Pfeifer, 1995) have hypothesized to solve the task following the decomposition and

integration approach. By fully exploiting the advantage of an intensive integration between the sensorimotor systems and the structure of the environment, the evolved individuals can overcome the problem posed by the fact that the two classes of stimuli are ambiguous in most cases. The classification of the two types of objects is not accomplished by a dedicated basic behavior (e.g. turning around the object until the classification is accomplished). Neither can we distinguish between a phase in which the individual should classify the object and a phase in which the classification has been accomplished. On the contrary, classification is an emergent property that is the result of the way in which the individual reacts to several different input stimuli.

4. Investigating the role of modularity

In the previous sections we showed how the decomposition-and-integration and the evolutionary approach may lead to qualitatively different ways of solving the same problem and how adaptation may be able to find simpler and robust solutions. In this section we shall investigate the role of modularity, that is the importance of having control systems organized into sub-components or modules.

From the point of view of the decomposition and integration approach, the use of modularity is implicit in the approach itself. The desired behavior is broken down into a set of basic behaviors or reflexes (such us “approach” or “avoid”) which correspond to a set of sub-components (modules) of the controller.

Conversely, in the case of the evolutionary approach the use of modularity is not mandatory. As we shown in the previous section, a non-modular control system such as a fully connected perceptron is perfectly able to produce a behavior that can be decomposed into a set of basic behaviors (in our case to explore the environment, to avoid walls, and to approach and remain close to targets).

Should we expect that behaviors of any complexity can be produced by homogeneous non-modular controllers or that modularity is necessary for certain tasks?

Moreover, should we expect a correspondence between modules and basic behaviors if we use modular controllers?

To answer these questions we will present a set of simulations in which neural controllers with different architectures have been trained using the evolutionary approach to produce a “garbage collecting behavior”.

4.1 Evolving a garbage collecting behavior

We decided to try to develop a control system for a Khepera 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.

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.

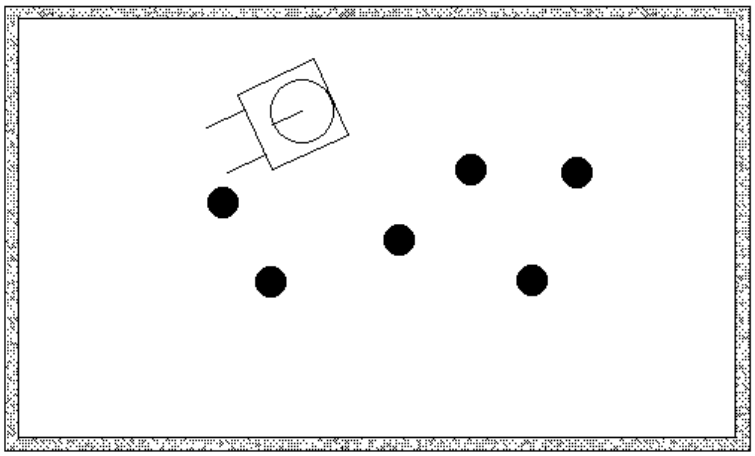

Figure 9. The robot and the environment.

In order to assess the role of modularity 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 simplest architecture used was a 2-layer feedforward neural network (see Figure 10a). The second architecture was also a feedforward neural network but had an internal layer of four units (Figure 10b). 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 10c). 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 10d) 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 10e) 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. In particular, the sub-division of the behavior into basic behavior corresponding to different neural modules is emergent.

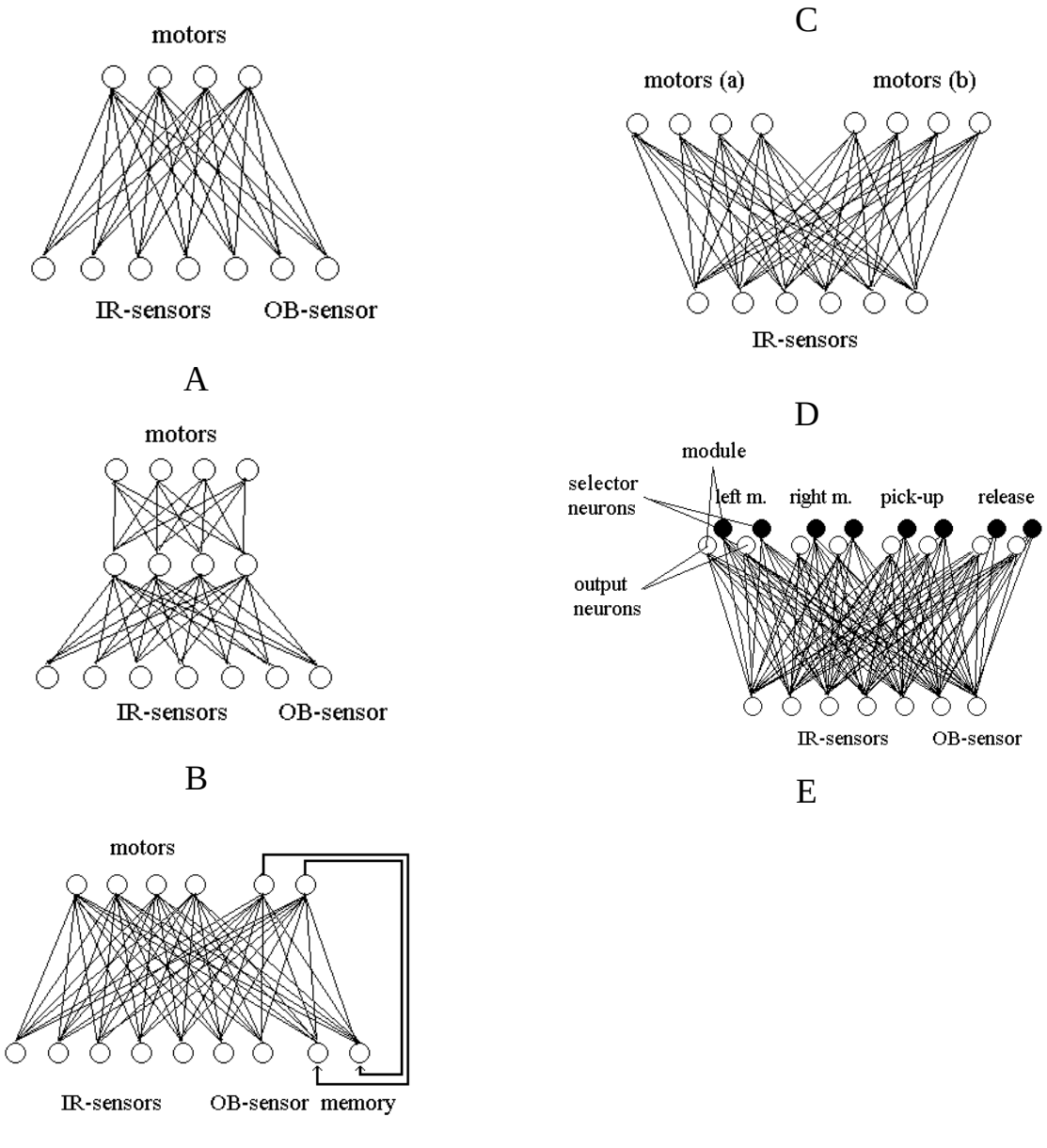

Figure 10. The 5 different architectures used to evolve the controller: (a) a standard feedforward architecture; (b) an architecture with an internal layer of hidden units; (c) a recurrent architecture; (d) a modular architecture with two pre-designed modules; (e) an emergent modular architecture.

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). Modular neural networks 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.

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). The process was repeated for 1000 generations.

The genetic encoding scheme was a direct one-to-one mapping. Genotypes had a total length of: (A) 256, (B) 416, (C) 480, (D) 480, and (E) 1024 bits in the 5 different architectures described.

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).

4.2 Emergent modularity speed-up evolution and allows the development of more robust controllers

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 11). 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 ($p < 0.05$).

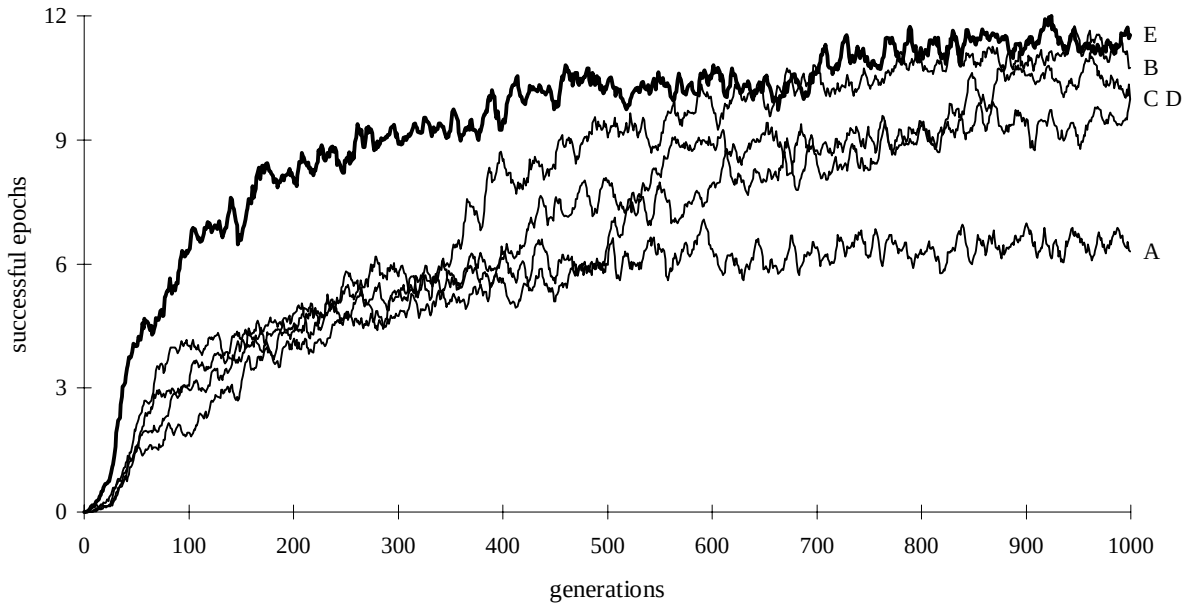

Figure 11. Number of epochs (out of 15) in which individuals with different architectures correctly picked up and then released a target object outside the arena through out generations. Each curve represents the average of the best individuals in 10 different simulations. Data smoothed by calculating 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 12). The best individuals of 7 (out of 10 simulations) 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.

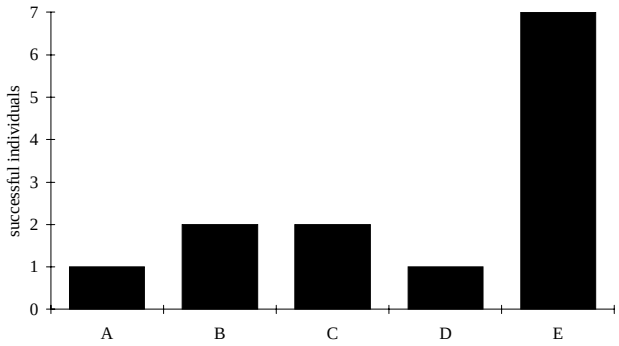

Figure 12. Number of evolved individuals for each control architectures capable of correctly picking up and then releasing outside the arena the 5 targets objects within 5000 cycles without displaying any incorrect behavior (e.g. crashing into walls, trying to grasp a wall, or trying to release a target over another target). A,B,C,D,E indicate the five different architectures described in Figure 10.

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.

4.3 There is no correspondence between evolved modules and basic behaviors.

At this point we may try to answer the second question we mentioned at the beginning of this section: can we find a correspondence between basic behaviors (e.g. to look for a target while avoiding walls; to look for a wall while avoiding targets etc.) and modules in evolved individuals?

If we analyze the type of solutions found by individuals with the architecture E which is the most successful architecture and which allows the evolutionary process to decide which module to use in any individual/environmental situations, we can say that the answer to this question is clearly no.

Figure 13 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 13, 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 basic 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).

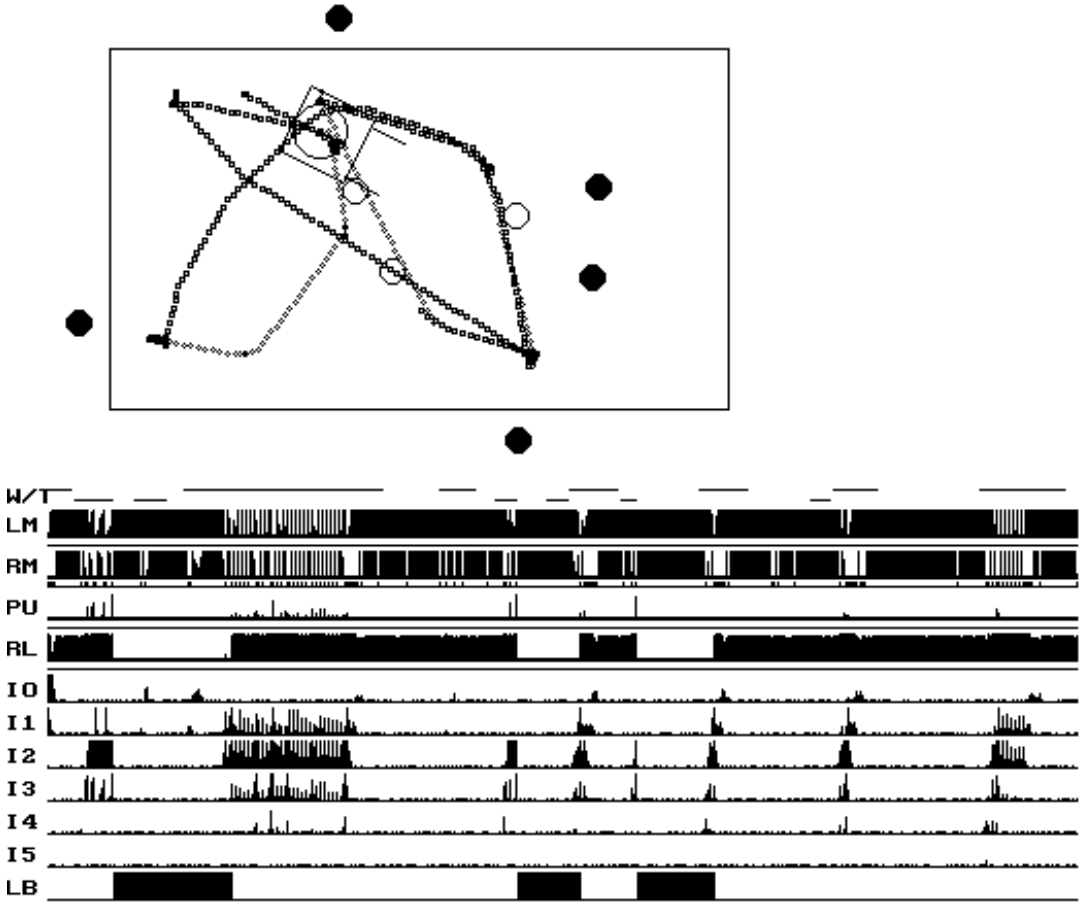

Figure 13. The top part of the figure represents the behavior of a typical evolved individual in its environment. Lines represent walls, empty and full circles represent the original and the final position of the target objects respectively, the trace on the terrain represents the trajectory of the robot. 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).

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 sub-behaviors. On the contrary each sub-behavior is the result of the contribution of different neural modules.

By observing the behaviors 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 13, 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.

Once again we have found that decomposition-and-integration and adaptation lead to qualitatively different ways of solving the same problem. While the decomposition and integration approach implies a correspondence between basic behaviors and modules, it is impossible to find such a correlation in evolved individuals which have modular neural controllers.

5. Discussion

We claimed that artificial evolution can produce more simple and robust solutions than a design process based on decomposition and integration. To support this hypothesis we analyzed the case of a Khepera robot that should classify objects of different shapes by remaining close to targets and avoiding walls. By comparing the solutions obtained by using the decomposition-and-integration and the evolutionary approaches we concluded that the latter approach allows more simple and robust solutions to be found (an extremely simple neural network controller resulted perfectly able to solve the task).

We also claimed that evolution and decomposition-and-integration lead to qualitatively different ways of solving the same problems. To support this hypothesis we showed how in the case of the task described in Section 3 the evolutionary approach leads to a solution in which the classification process is an emergent property of the dynamics between the individual behavior and the environment. On the contrary, the decomposition and integration approach leads to solutions in which the classification process is accomplished by a basic behavior dedicated to this purpose and implemented in a separate sub-component of the controller.

We have also shown how in the case of the more complex task described in Section 4, evolution leads to a qualitatively different solution from that obtained using the decomposition and integration approach. Modularity allows evolution to produce individuals with the required ability more quickly and to discover more robust solutions. However, while the decomposition and integration approach implies a correspondence between basic behaviors and modules, it is impossible to find such a correlation in evolved individuals which have modular neural controllers. Once again the two approaches lead to qualitatively different ways of solving the same problem. In the case of the decomposition and integration approach modularity is used to reduce the complexity of the controller parts to be designed or learned. In the evolutionary approach instead, it is used to allow the controller to produce very different motor responses in very similar sensory situations.

5.1 Evolution operate at a lower level than decomposition and integration

At this point we may try to understand why this happens. Why evolution leads to qualitatively different types of solutions from decomposition and integration. Moreover why, at least in the two cases discussed in this paper, evolution allows very simple solutions to be found.

To understand the differences between evolution and decomposition-and-integration we have to distinguish, as proposed by Sharkey and Heemskerk (in press), 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 sensorimotor loops (*distal description of behavior*) and a description from the point of view of the agent's sensorimotor system that accounts for how the agent itself reacts in different sensory situations (*proximal description of behavior*).

Artificial evolution operates at the level of the proximal description of behavior while decomposition-and-integration imposes constraints at the level of the distal description of behavior.

Consider the task of classifying objects with different shapes discussed in Section 3. In the evolutionary approach discussed in section 3.2 evolution selects the weights of the neural networks that determine the speed of the two motors for each possible state of the 6 sensors (i.e. mapping between the sensory space and the motor space). A sequence of input-output mappings, given a certain initial position of the robot and a particular environment, produce a sequence of movements that can be described, at the distal level, as basic behaviors (e.g. "avoid a wall" or "keep close to a target"). Evolution operates at the level of the proximal description of behavior by selecting the weights of the neural controllers which in turn determine the mapping between sensory stimuli and motor actions. Individuals are selected on the basis of their ability to accomplish the whole task. Distal descriptions of behavior do not play any role in the evolutionary process (see Figure 14).

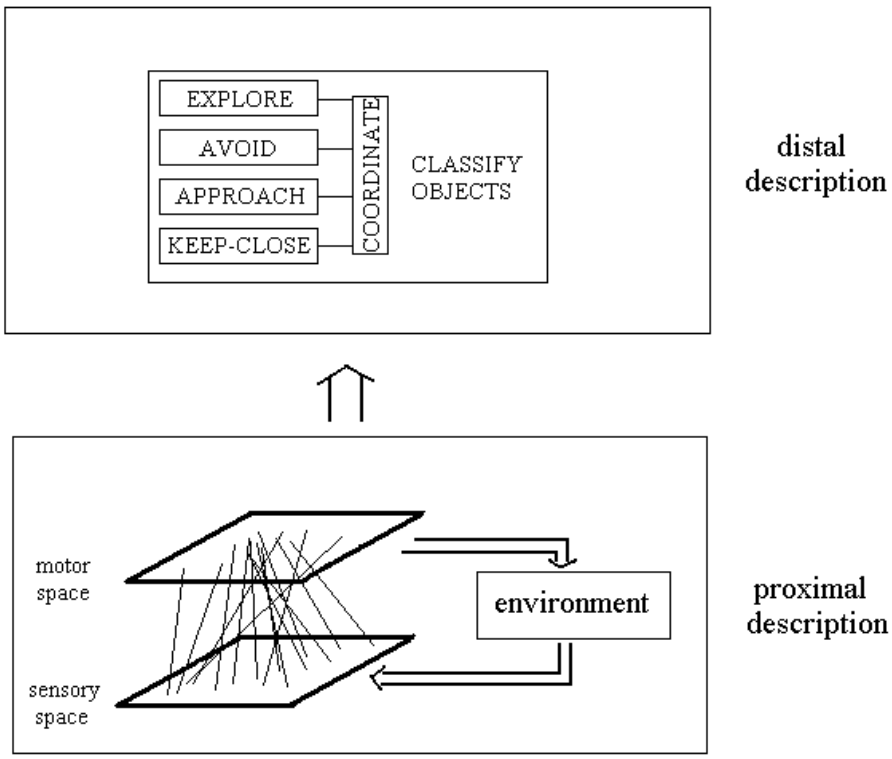

Figure 14. Proximal and distal descriptions of behavior for the task described in Section 3. In the case of the evolutionary approach, the basic behaviors (on top) are distal descriptions of sensorimotor sequences. Such sequences are the result of the control system that maps sensory states into motor states and of the environment that co-determines, together with the motor output of the individual, the next sensory stimulus of the individual itself.

In the case of the decomposition and integration approach, on the contrary, the distal description of behavior plays a role in the very first part of the design; the sub-division of the problem into basic behaviors to be coordinated. A learning mechanism can then be used to train the different modules to perform the corresponding expected basic behaviors (Dorigo and Schnepf, 1993) and/or to coordinate them (Mahadevan and Connell, 1992). However the learning process is limited by boundaries imposed by the distal description of the desired behavior.

A similar situations can be observed in the second experiment described in Section 4. Also in this case evolution operates at the level of the proximal description of behavior by selecting the weights of the neural controller which in turn determines the mapping between sensory stimuli and motor actions. Individuals are selected on the basis of their ability to release objects outside the arena and for their ability to have objects in the gripper (for a discussion on why this second criterion is necessary see Nolfi, in press). In this case individuals have the possibility to use different modules for producing different parts of the required sensorimotor mapping and they actually exploit this possibility by dividing the sensorimotor mapping into sub-parts managed by different sub-networks (see Figure 15). To exploit modularity thus evolution operates a decomposition and integration process. However this process takes place at the level of the proximal description of behavior. As a consequence no correspondence can be found by subdividing the sensorimotor mapping into parts and by subdividing the whole behavior into basic behaviors as shown in Section 4.3.

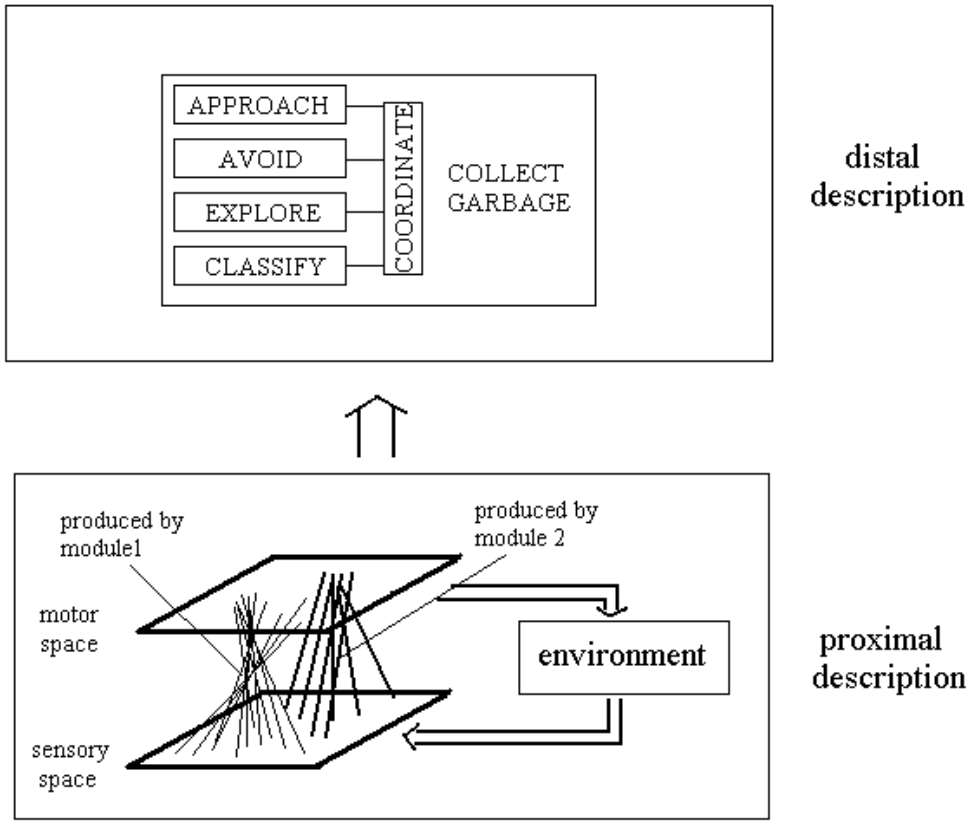

Figure 15. Proximal and distal descriptions of behavior for the task described in Section 4. Distal descriptions of sensorimotor sequences are represented at the top of the Figure. Such sequences are the result of the control systems that maps sensory states into motor states and of the environment which co-determine, together with the motor output of the individual, the next sensory stimulus of the individual itself. Sensorimotor mapping is accomplished by different sub-parts of the controllers. Both the mapping and the sub-division into different components are subjected to the evolutionary process.

5.2 Evolution reduces the complexity of the controller by exploiting the complexity of the environment.

The previous section we attempted to explain why evolution and decomposition-and-integration usually lead to qualitatively different solutions. In this last Section we will try to explain how evolution can lead to simple solutions.

There are probably more than one answer to this question and we do not claim to be exhaustive here.

A very general answer is the following: 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. As clearly shown by Braitenberg (1984) the complexity of the behavior of an agent can largely be due to the complexity of the environment and not of the agent itself. Evolution, by operating at the level of the close description of behaviors, can shape individuals so to allow them to use the complexity of the environment to produce behaviors that are complex at the level of the distal description.

A clear example of this is the solution that our evolved individuals find to keep close to the target. As we describe in Section 3 individuals, at a distal level of description, show a dynamic behavior: they move back and forth close to the target. However, these individuals cannot produce a dynamic behavior by themselves because they have simple sensorimotor controllers that cannot keep trace of the past sensory stimuli or of past actions they have performed. They behave in a way that, by exploiting the complexity of the environment which co-determine their next sensory state, can be described as a dynamical behavior from a distal description point of view.

Another good example is the way in which the evolved individual described in Section 3.3 classifies objects of different types by avoiding walls and approaching targets. As we said this individual is able remain close to targets, only approaching them from about 30 to 60 degrees on the right side. At the same time this individual is able to approach targets from this angle about 30% of the times. Conversely, he is able to approach walls from the left side (the side from which he quickly avoids any abject) about 70% of the times. In other words this individual exploits the regularities of the environment to increase the frequency of the favorable sensory situations. It behaves in way that maximizes the probability of encountering a target between 30 and 60 degrees on the right (i.e. the angular interval from which he is able to approach it and to remain close to the target) and maximizes the probability of encountering a wall from the left side (i.e. the side from which it quickly avoids any object it encounter).

References

Braintenberg, 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, pp.14-23.

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

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

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

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

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

Mondada, F., Franzi, E., and 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.

Nolfi, S., Florano D., Miglino, O., and Mondada, F. (1994). How to evolve autonomous robots: different approaches in evolutionary robotics. *Artificial Life IV, Proceedings of fourth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge MA, MIT Press.

Nolfi, S. (in press). Evolving non-trivial behaviors on real robots: a garbage collecting robot. *Journal Robotics and Autonomous System*, special issue on "Robot learning: The new wave"

Nolfi, S., Miglino, O., and 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.

Rumelhart, D.E., Hinton G.E., and Williams, R.J. (1986). Learning internal representations by error propagation. In D.E. Rumelhart, and J.L. McClelland (Eds.) *Parallelel Distributed Processing*. Cambridge, MA, MIT Press.

Scheier, C., and Pfeifer, R. (1995). Classification as sensorimotor 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, pp.862-875.

Scheier, C., and Lambrinos, D. (1996).Adaptive Classification in Autonomous Agents. In: Trappl, R. (ed.), *Cybernetics and Systems '96 Proceedings of the 13th European Meeting on Cybernetics and Systems Research EMCSR 96*, Vienna, Austria, pp. 1037-1043

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