
**Institute of Psychology
C.N.R. - Rome**

**Evolving non-Trivial Behaviors on Real Robots:
a garbage collecting robot**

Stefano Nolfi

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

March 1996 (revised December 1996)

Technical Report 96-04

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

To appear on: Journal Robotics and Autonomous System, Special Issue on "Robot learning:
The new wave".

Evolving non-Trivial Behaviors on Real Robots: a garbage collecting robot

Stefano Nolfi

Institute of Psychology, National Research Council
15, Viale Marx - 00187 - 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/nolfi.html>

Abstract

Recently, a new approach involving a form of simulated evolution has been proposed to build autonomous robots. However, it is still not clear if this approach is adequate for real life problems. In this paper we show how control systems that perform a non-trivial sequence of behaviors can be obtained with this methodology by "canalizing" the evolutionary process in the right direction. In the experiment described in the paper, a mobile robot was successfully trained to keep clear an arena surrounded by walls by locating, recognizing, and grasping "garbage" objects and by taking collected objects outside the arena. The controller of the robot was evolved in simulation and then downloaded and tested on the real robot. We also show that while a given amount of supervision may canalize the evolutionary process in the right direction the addition of unnecessary constraints can delay the evolution of the desired behavior.

1. Introduction

Recently, a new approach to the development of autonomous robots based on an automatic design process has been proposed. It has been called *Evolutionary Robotics* because it involves a form of artificial evolution (Cliff, Harvey, and Husband, 1993).

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

Husband, 1993; Nolfi, Floreano, Miglino and Mondada, 1994).

Researchers interested in this approach are currently investigating six main issues (for a review see Matarik, and Cliff, 1996): (a) the feasibility of the use of simulation in order to speed up the evolutionary process (Brooks, 1992; Nolfi, Floreano, Miglino, and Mondada, 1994; Jacobi, Husbands, and Harvey, 1995; Miglino, Lund, Nolfi, 1995); (b) the importance of introducing a mapping process between the genotype and the phenotype that could capture at least some of the important properties of the ontogenetic developmental process in natural organisms (Cliff, Husband and Harvey, 1993; Nolfi, Miglino, and Parisi, 1994; Gruau, 1995); (c) the integration of a lifetime learning process with the evolutionary process in order to enhance the adaptation power of the genetic algorithm (Floreano, and Mondada, 1996b) or in order to allow adaptation to rapid changes in the environment (Nolfi, and Parisi, in press); (d) the evolution of the hardware architecture (Higuchi et al. 1992; Thomson 1995); (e) The evolution of social behaviors (Matarik, 1994; Vanio et al.,

1995); (f) the attempt to apply this methodology to solve relatively complex tasks (Nolfi, and Parisi, 1995). In this paper we will focus on this last topic. We will show how control systems that perform a non-trivial sequence of behaviors can be obtained with this methodology by carefully designing the conditions in which the evolutionary process operates.

2. Related Work

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

Only recently has the evolutionary approach produced results that have been validated on real robots. Lewis, Fagg and Sodium (1992) evolved a motor controller for a six-legged robot called Rodney that was able to walk forward and backward. Colombetti and Dorigo (1992) have evolved a control system for a robot called Autonomouse to perform a light approaching and following behavior. Several authors have reported experiments in which a Khepera robot was trained to perform an obstacle avoidance task (Floreano and Mondada (1994); Nolfi, Floreano, Miglino, and Mondada (1994); Jacobi, Husbands, and Harvey, 1995; Miglino, Lund, Nolfi, in press). Yamauchi and Beer (1994) describe an experiment in which recurrent neural networks are evolved to solve a landmark recognition task using a sonar. They tested the network on a Nomad 200 robot with built-in wall-following behavior. Harvey, Husband, and Cliff (1994) evolved a system capable of approaching a visual target (in the most complex experiment the system was successfully trained to approach a triangle target and to distinguish it from a rectangular one). The system was not implemented on a standard autonomous robot but on specially designed robotic equipment in which the robot

was suspended from a platform which allowed translational movements in the X and Y directions. Miglino, Nafasi, and Taylor (1995) evolved a controller for a mobile Lego robot that should explore an open arena. Finally, Floreano and Mondada (1996a) evolved a controller for a Khepera robot that performed an homing navigation.

In some of these experiments the evolutionary process was conducted in simulation and then the control system obtained was downloaded and tested on the robot (Colombetti and Dorigo, 1992; Yamauchi and Beer, 1994; Miglino, Nafasi, and Taylor, 1995; Miglino, Lund, Nolfi, 1995; Jacobi, Husbands, and Harvey, 1995). In other cases the evolutionary process was conducted entirely on the real robot (Lewis, Fagg, and Sodium, 1992; Floreano and Mondada, 1994, Harvey, Husband, and Cliff, 1994; Floreano, and Mondada, 1996a). In other cases evolution took place in part in simulation and then it was continued on the real robot (Nolfi, Floreano, Miglino, and Mondada, 1994). When the evolutionary process was conducted partially or totally on the real robot, in most of the cases, the evaluation process was conducted automatically, i.e., without requiring an external support while in the case of Lewis et al. performance was evaluated by human observers. In all of the work described neural networks were used in order to implement the controller, with the exception of that of Colombetti and Dorigo (Colombetti and Dorigo, 1992) who used classifier systems.

This work clearly shows that evolutionary robotics is a very active and promising new field of research. However, it is also clear that real life problems require a system capable of producing behaviors significantly more complex than those described above.

3. Our Framework

Having at our disposal a Khepera robot with the gripper module (see next section) we decided to try to develop a control system for a robot that should keep clear an arena surrounded by walls. The robot has to 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) to explore the environment, avoiding the walls; (b) to recognize a target object and to place the body in a relative position so that it can be grasped; (c) to pick up the target object; (d) to move toward the walls while avoiding other target objects; (e) to recognize a wall and place the body in a relative position that allows the object to be dropped out of the arena; (g) to release the object. Moreover, these sub-tasks can be broken down into smaller components. For example (a) may be broken down into (a₁) go forward when sensors are not activated; (a₂) turn left at a given speed when right sensors are activated etc. However, we want the complete solution to the task to emerge through an evolutionary process and therefore we do not need to specify the requested behavior in detail or to analyze the interferences between sub-behaviors.

Scheier and Pfeifer (1995) developed the control systems for a Khepera robot that performs a task very similar to that 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 its home base. However they decided to program by hand a set of elementary behaviors (move forward, turn toward objects, avoid obstacles, grasp, and bring to the nest) and let them all run in parallel in order to obtain the desired behavior. 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 that correspond to small pegs with the grasp behavior; in other words to classify the two types of pegs. On the contrary we want the entire behavior and its organization into sub-behaviors to emerge during the evolutionary phase.

Colombetti, Dorigo, and Borghi (1996) also studied a similar task. 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.

Given our previous experience with Khepera and the difficulties of evolving a behavior of this type in the real robot we decided to conduct the evolutionary process in simulation by using an extended version of our Khepera simulator described in Nolfi, Floreano, Miglino, and Mondada, 1994. The control system obtained was then downloaded into the robot and tested in the real environment. In this section we will describe the robot, the environment, and the simulator. In section 4 we will describe the architecture of the controller and the type of genetic algorithm and fitness formula used. In section 5 we will describe the results obtained in simulations and on the real robot. Finally, in section 6, we will discuss the amount of supervision required to allow evolution to discover a complete solution to the task.

3.1. The Robot

The robot was Khepera (Figure 1), a miniature mobile robot developed at E.P.F.L. in Lausanne, Switzerland (Mondada, Franzi, and Jenne, 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.

3.2. The Environment

We built a rectangular arena of 60x35 cm surrounded by walls which contains 6 target objects. The walls were 3 cm in height, made of wood, and were 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.

3.3. The Simulator

To evolve the controller of the robot in the computer the simulator described in Nolfi, Floreano, Miglino, and Mondada (1994) was

extended in order to take into account the gripper module of Khepera.

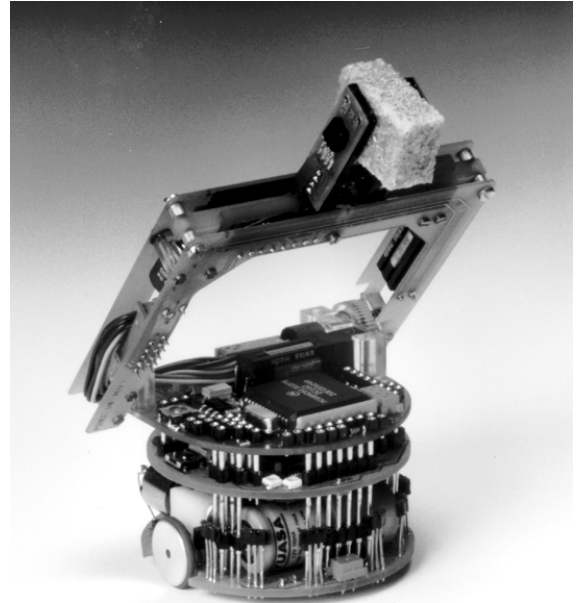


Fig.1. The Khepera robot.

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°, 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.

4. Evolving the Controller

Like the majority of people who use evolutionary methods to obtain control systems for autonomous robots we decided to implement the controller with 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 able to generalize their ability in 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 made by the human designer (Cliff, Harvey, and Husband, 1993) and synaptic weights and neurons are 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 Litmann, 1991; Nolfi, Elman and Parisi; 1994). In the following sections we will describe the architecture, the fitness formula, and the form of genetic algorithm used.

4.1. The Neural Controller

We tried several different network architectures (see below) and found that the best architecture was a feedforward network with 7 sensory neurons, 16 motor neurons, and no internal neurons. 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 (see Figure 2). On the motor side we had four pairs of motor neurons that codifies the speed of the left and right motors and the triggering of the "object pick-up" and "object release" procedure respectively and four pairs of selector neurons that determined which of the two competing output neurons got the control of the corresponding robot's actuator each time step (the competitor with the corresponding highly activated selector neuron gets control). This architecture allows modular solutions (i.e. solutions in which different parts of the nervous system are used in different situations) to emerge without pre-defining the correspondence between behaviors and modules as is required in sub-sumption architectures (Brooks, 1986). The other architectures that turned out to be less effective were: (1) a simple architecture with just 7 sensory and 4 motor neurons; (2) an architecture with two additional recurrent neurons; (3) an architecture with an additional internal layer with 4 neurons; (4) an architecture with two sets of 4 output neurons (i.e. with two pre-defined modules), one for the target finding and pick-up behavior and one for the wall approaching and object release behavior (for a systematic comparison see Nolfi, in press).

It is important to note that the task chosen 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 to the type of the object; wall or target (in the two cases the infrared sensors have only slightly

different activation values). This may explain why 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.

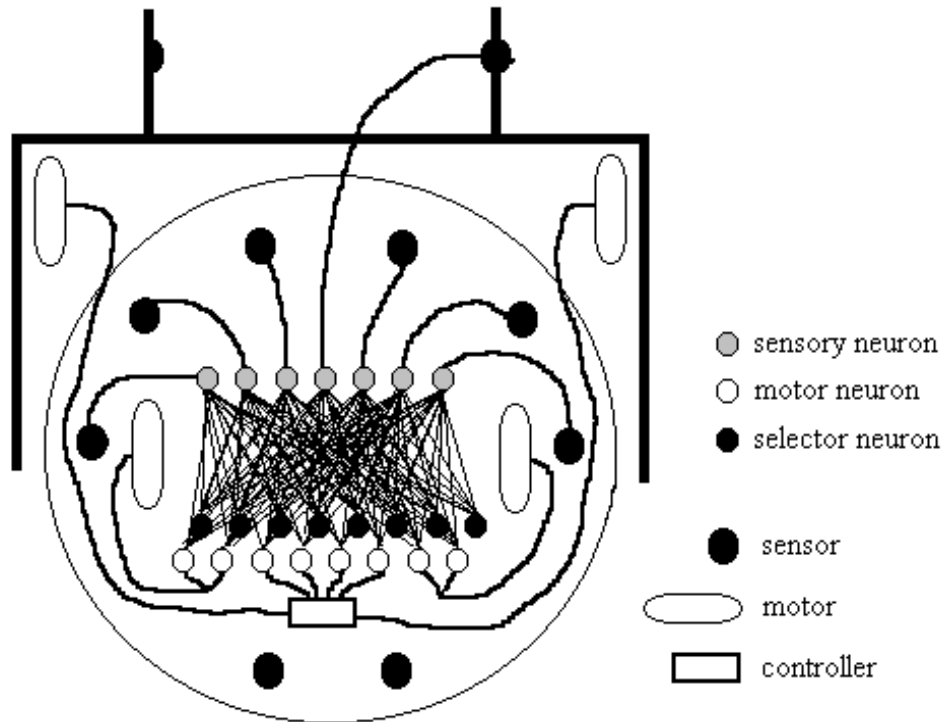


Fig. 2. The control system of the robot. The 7 sensory neurons represented with gray small circles are directly connected to the 6 frontal sensors of the robot located on the circular body of Khepera and to the light barrier sensor located on the gripper. Two pairs of motor neurons represented with empty small circles were connected to the two motors that controlled the wheels of Khepera while the other two were connected to the two motors of the gripper through a controller. The four pairs of selector neurons represented with full small circles determine which of the two corresponding output neurons got control each time step (the output neuron with the corresponding highly activated selector neuron gained control).

The activation of the sensors and the state of the motors are encoded each 100 milliseconds. However, when the activation level of the "object pick-up" or of the "object release" neurons reach a given threshold a sequence of action occurs that may require 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).

The activation values of the infrared sensors (which can have 1024 different values ranging from 0 to 1023) and 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).

To accomplish the task the weights of the neural network should be set in such a way that Khepera can perform the following sequence of behaviors:

- explore the environment, avoiding the walls
- recognize a target object and place the body in a relative position that allows it to be grasped
- pick up the target object
- move toward the walls while avoiding other target objects
- recognize a wall and place the body in a relative position that allows the object to be dropped out of the arena
- release the object

4.2. The Genetic Algorithm

To evolve neural controllers able to perform the task described above we used a form of genetic algorithm. We began with 100 randomly generated genotypes each representing a network with the architecture described in the previous section and a different set of randomly assigned connection weights. This is Generation 0 (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. 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). Random mutations were introduced in the copying process, resulting in possible changes of the connection weights (all 100 individuals were mutated). 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. The encoding scheme is the way in which the phenotype (in this case the connection weights of the neural network) is encoded in the genotype (the representation on which the genetic algorithm operates). The one-to-one mapping is the simplest encoding scheme in which one and only one 'gene' corresponds to each phenotypical character. (For more complex encoding schemes also allowing evolution of the neural architecture, see Cliff, Harvey and Husband, 1993; Nolfi, Miglino, and Parisi, 1994). In our case, to each of the 128 parameters (112 connection weights and 16 biases) corresponds a sequence of 8 bits in the genotype which has a total length of 1024 bits. Normal binary encoding was used to translate the 8 bits to one weight value between -10.0 and + 10.0. Connection weights and biases are fixed (i.e. there is not learning during individual's lifetime).

4.3 The Fitness Formula

The fitness formula is the way in which individuals are evaluated in order to decide who is allowed to reproduce.

In our case, the simplest thing would be to score individuals by counting the number of objects correctly released outside the arena. However, the probability that a network with random weights could succeed in doing the complete sequence of correct behaviors (i.e. to find, grasp and bring out of the arena an object) even once is extremely low, all networks of the initial generations would be scored with the same 0 values and, as a consequence, the selection process would not have any affect.

In order to avoid this problem one can use a more complex fitness formula that allows individual differences in behavior to be captured also in initial generations. We used a fitness formula with 2 components and in addition we exposed the robot to useful learning experiences (see below). This meant that individuals were scored not only for their

ability to perform the complete sequence of correct behaviors but also for their ability to perform portions of the complete sequence. In particular, we increased the fitness of an individual in the following cases:

- if the robot had an object in the gripper
- if the robot released the object outside the arena

In addition to add a fitness component, we found it important to expose the robots to useful learning experiences during the evolutionary process in order to allow the evolutionary process to discover a solution to the problem. In particular, we found it important to increase the number of times 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 robots each time they picked up a target (i.e. by controlling the learning experiences of the robot).

Individuals were scored with 1 for each cycle they had an object in the gripper and with 2000 for each object correctly released outside the arena (an high score was used in order to force the evolutionary process to select individuals able to performe the entire sequence of behaviors).

5. Results

We run 10 simulations starting with populations of 100 networks with randomly assigned connection weights. Each simulation lasted 1000 generations (about 10 hours using a standard IBM RISC/6000). The evolved neural controllers were then downloaded on to the real robot and tested in the real environment. In the next sections we describe the results obtained in the simulated and real environments.

5.1. Performance in the simulated environment

If we measure the number of epochs (out of 15) in which individuals: pick up a target, pick up and correctly release a target, or fail

(by crashing into a target or a wall) throughout generations we can see in Figure 3 that, in the very first generations, individuals are able to pick up target objects only occasionally while most of the time they crash into them or into walls. However, in the next 50 generations, the number of crashes significantly decreases, the number of targets picked up increases, and individuals that are able to release the targets outside the arena, even if only occasionally, are selected. Later on, individuals that are more and more able to correctly pick up targets and to release them out of the arena are selected. After 1000 generations individuals are able to perform the entire sequence of correct behaviors 11.8 out of 15 epochs on average, and 15 out of 15 epochs in the case of the best individual of the most successful simulation.

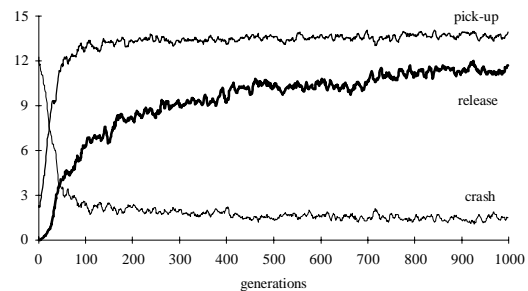


Figure 3. Number of epochs out of 15 in which individuals: *crash* into walls or other targets; *pick-up* an object but do not release it; *pick-up* and then *release* an object outside the arena. Average of the best individuals of each generation for 10 replications of the simulation. Data smoothed by calculating rolling averages over preceding and succeeding 3 generations.

Figure 4 represents the behavior of a typical evolved individual and the corresponding state of the motor and sensory neurons throughout 600 cycles. The robot, which started from the top-right part of the arena, after correctly picking-up and releasing 5 targets, is approaching the last target still in the arena. As can be seen from the trace on the terrain and at the activation state of the motors, when the robot perceives something, it starts to move back and forth for a variable amount of time until it recognize the perceived object. It then acts consequently by picking-up objects and avoiding walls if the gripper is empty and by

otherwise avoiding targets and releasing the object outside the wall. It should also be noted that during the back and forth phases the robot also modifies its orientation to correctly recognize the perceived object and at the same time to place itself in the right position in

order to pick up or release a target or in order to avoid a wall or a target, depending on the circumstances (for more details on how the robot classify the two type of objects see Nolfi, 1996).

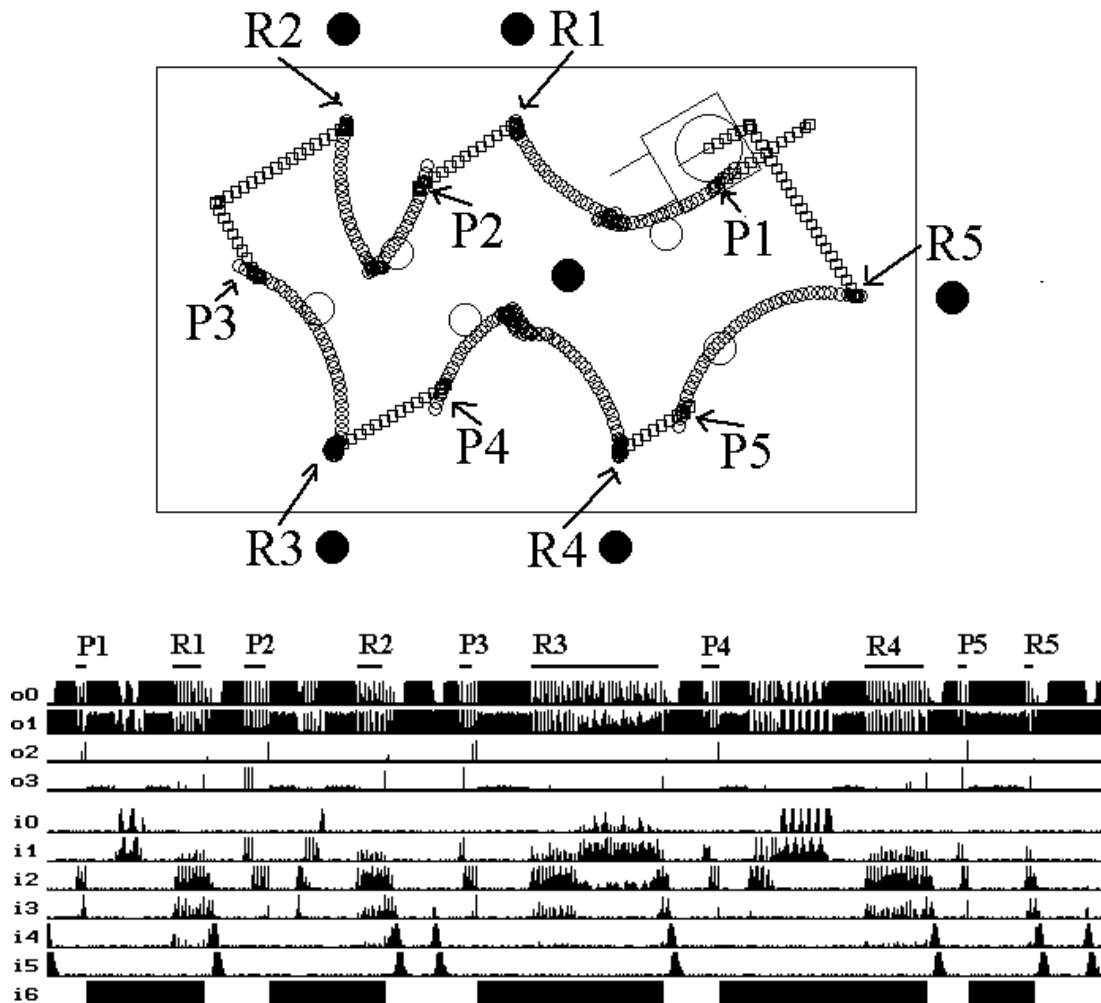


Figure 4. The top part of the figure represent the behavior of a typical evolved individual. Lines represent walls, empty and full circles represent the original and the final position of the target objects, respectively, finally the trace on the terrain represents the trajectory of the robot. The bottom part of the figure represents the state of the four motor neurons that have control of the motors and of the 7 sensors (the 6 infrared and the light-barrier sensor, respectively) throughout time. P1 to P5 and R1 to R5 indicate the phases in which the individual picked-up and released the 5 corresponding targets. Between P1 and R1 and between P4 and R4 the reader can recognize two phases in which the robot avoided the target positioned in the middle while carrying another target.

5.2. Performance in the real environment

We downloaded on Khepera the neural controllers of the best individual of each simulation (i.e. the individuals of the last generation with the best performance in the simulated environment) and tested them in the real environment.

Figure 5 shows a comparison between the performance in the simulated and real environment for the best individuals of the last generation for each simulation. As can be seen, in most cases a decrease in performance is observed in the real environment. However, all individuals were able to perform the entire sequence of actions correctly at least 40% of the time. The best individual (i.e. the best individual of simulation 6) was able to correctly pick-up targets 93% of the time, it always released targets correctly while avoiding other targets, never crashed into the walls, and never tried to incorrectly grasp the walls.

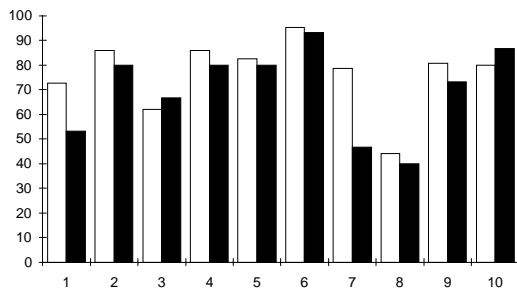


Figure 5. Percentage of times in which the entire sequence of behaviors (to find, to pick up, and to release a target outside the arena) is accomplished correctly in the simulated and real environment. Empty and full histograms respectively represent performance in the simulated and real environment of the best individual of the last generation for 10 replications of the simulation. Each individual was evaluated for 150 trials in the simulated environment and for 15 trials in the real environment.

By letting the 10 best individuals of the last generation free to interact with the real environment for 5000 cycles we found that 7 out of 10 individuals were able to completely clean the arena by removing 5 target objects. 6 of these individuals were able to accomplish

the task without showing any incorrect behavior (e.g. to crash into the walls, to try to grasp a wall or to release a target over another target). The best individual was able to completely clean the arena in 761 cycles (about 1.5 minutes). Given this successful performance it did not seem necessary to continue the evolutionary process in the real environment to allow individuals to adapt to the differences between the simulated and the real environment (see Nolfi, Floreano, Miglino, and Mondada, 1994).

The ability to collect targets generalized also to objects with different dimensions and shape with respect to those used in the training phase (i.e. cylinders with a diameter of 2.3cm). Evolved individuals were also able to collect and correctly depose of objects that were both larger and smaller in diameter than the objects encountered during training.

6. Emergence versus supervision

In principle, artificial evolution can work with no supervision at all beside a criterion for evaluating how much evolving individuals accomplish the desired task. The same framework can be applied in order to evolve simple or very complex tasks. However, as we saw in section 2, up to now only relatively simple tasks have been accomplished with this methodology. This limited result contrasts with the extraordinary large number of life forms, most of them far more complex than any human artefact, produced by natural evolution. One can argue that natural evolution has had at its disposal huge resources and an enormous amount of time while experiments in artificial evolution last only a few hundred generations and are carried on in very simple environments. However, people who work in this field know that, in a typical experiment, after one or few hundred generations, artificial evolution stabilizes on a desirable or undesirable state and nothing new happens.

The lack of power of current models using artificial evolution can be explained in several ways. Certainly, it could be that these models lack some important property of natural evolution. For example it could be that the way in which genetic information is

represented and translated into the corresponding phenotype is inadequate. However, we think that there is another important reason: using a very specific selection criterion implies that the evolutionary process is used in at least partially distorted way; as an optimization technique. Natural evolution does not optimize any specific competence in addition to reproductive ability. It discovers competences because they can enhance reproductive success. However, this is accomplished without any specific constraint. In natural organisms, complex competencies (like the ability to fly or to build a nest) are acquired by natural organisms by modifying pre-existing competencies acquired for other reasons (Gould, 1991). As a consequence, it is not surprising that specific complex behaviors are difficult to obtain through artificial evolution if one starts from scratch.

How we can overcome this problem if we want to develop artefacts that have specific competencies? There are two possibilities; one is to imitate nature, leave the evolutionary process free to discover competencies indirectly useful for a very general purpose (such as the ability to reproduce) and hope that the desired competencies emerge; the second is to try to force evolution in the right direction by favouring the emergence of competencies that in turn can favour the emergence of the required competencies. In both cases emergence would play an important role but in the second case a certain amount of supervision would be introduced in order to "canalize" the evolutionary process.

Constraints that canalize the evolutionary process can be introduced in several ways: by favouring the emergence of basic competencies with the introduction of additional components in the fitness function, by controlling the type of experiences to which individuals are exposed during the evolutionary phase, and by modifying the body of the individuals or their internal organization (i.e. internal architecture). In the case of the task discussed in this paper, in order to obtain the ability to keep the arena clean, we found it necessary:

(a) to add an additional component in the fitness function (a reward for the ability

- to pick up objects) to favour the emergence of the corresponding ability.
- (b) to increase the frequency of the stimuli corresponding to the situation: "object in front of the robot and object on the gripper" to force the robot to learn to avoid targets while carrying other targets.
- (c) to use an emergent modular architecture for the control system to allow a modular solution to be selected by the evolutionary process.

These constraints proved to be necessary experimentally because, by removing one or more of them, a significant decrease, on average, in the ability to perform the garbage collecting task was observed. It is interesting however to note that a loss in performance is observed also on adding additional constraints. In the first simulations we did, we used a fitness formula with 10 components. Later on we realised that 7 of them not only were unnecessary but actually retarded the evolution of correct behaviors. The 7 non useful components were as follow: (1) a reward when the robot is close to the target object; (2) a reward if the target object is in front of the robot; (3) a reward if the robot tries to pick up the object; (4) a reward if the robot, when it has an object on the gripper, gets close to a wall; (5) a reward for the ability to bring objects out of the walls; (6) a punishment when the robot crashes into walls while carrying an object; (7) a punishment when the robot releases the object on to another target.

Figure 6 shows the number of targets correctly released outside the arena throughout generations for simulations with 1, 3, and 10 components (average results for 10 replications). As can be seen, optimal performance is obtained using the fitness formula described in section 4.3 which has 3 components. By removing the two additional components, individuals never acquire the ability to perform the task. By adding further 7 components to the fitness formula, a delay in the emergence of the required ability is observed in the first 200 generations.

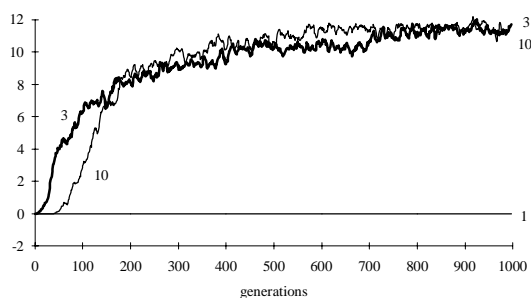


Figure 6. Number of epochs (out of 15) in which individuals pick up and then release a target object outside the arena for fitness formulae with 1, 3, and 10 components throughout generations. Each curve represents the average of the best individuals of each generation for 10 replications of the simulation. Data smoothed by calculating rolling averages over preceding and succeeding 3 generations.

7. Discussion

Intelligent autonomous robots should be able to use sensory information to behave in an external environment. As the environmental complexity grows and the task becomes more complex, the design of the system becomes more and more difficult. To overcome this problem the *behavior-based* approach proposed by Brooks (1986) uses the idea of designing several simple sensory-motor processes and a coordination technique. However, one can argue that even the design of the simple base behavior may result difficult (see Nolfi, 1996) and that the co-ordination between them may become immanageable as soon as the number of required basic behaviors increase. Evolutionary robotics allows to overcome these problems by letting the entire behavior of the system, including its organization into basic behaviors and the co-ordination between them, emerge through a process based on selection and differential reproduction (see Nolfi, in press).

In principle, by using this approach, the design process may be reduced to the choice of the criterion for evaluating to what extent evolving individuals accomplish the desired task. However, we showed that, in the case of relatively complex tasks like that presented in this paper, this is insufficient. Some additional intervention is needed to canalize the evolutionary process in the right direction.

This canalization process can be accomplished in several ways: by adding components to the fitness formula that may favour the emergence of competencies that in turn can favour the emergence of the required ability; by manipulating the type of stimuli individuals experience during the evolutionary phase, by choosing the architecture of the control system etc.

We also showed that the amount of canalization pressure should be kept as small as possible. The addition of unnecessary constraints can delay (and even prevent) the evolution of the desired behavior. It is interesting to note for example, that the emergent modular architecture, which allows the organization into sub-behaviors and their co-ordination to emerge during the evolutionary phase, outperforms the architecture in which the organization into sub-behaviors is designed by hand. This despite the emergent modular architecture requires a double number of weights and as a consequence implies that a much larger space has to be searched by the genetic algorithm.

References

- Ackley D. H. and M. L. Littman (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).
- Brooks R. A. (1986). A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation* 2, 14-23.
- Brooks R. A. (1992). Artificial life and real robots, 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).
- Cliff D. T., I. Harvey and P. Husbands. (1993). Explorations in Evolutionary Robotics. *Adaptive Behavior* 2, 73-110.
- Colombetti M. and Dorigo M. (1992). Learning to control an autonomous robot by distributed genetic algorithms, in: J. A. Meyer, H. L. Roitblat, and S. W. Wilson, (eds.), *From Animals to Animats 2*, Proceedings of 2rt International Conference on Simulation of Adaptive Behavior (Cambridge, Mass., MIT Press).

- Colombetti M., Dorigo M., and Borghi G. (1996). Behavior analysis and training. A methodology for behavior engineering. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, (26) 3, 365-380.
- Floreano D. and Mondada F. (1994). Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot, in: D. Cliff, P. Husbands, J. Meyer, S. W. Wilson, (eds.), *From Animals to Animats 3: Proceedings of Third Conference on Simulation of Adaptive Behavior* (Cambridge, Mass, MIT Press/Bradford Books).
- Floreano D. and Mondada F. (1996a). Evolution of homing navigation in a real mobile robots, *IEEE Transaction on Systems, Man, and Cybernetics. -Part B: (26) , 3*, 396-407.
- Floreano, D. and Mondada, F. (1996b). Evolution of Plastic Neurocontrollers for Situated Agents. In P. Maes, M. Mataric, J-A. Meyer, J. Pollack, and S. Wilson. (Eds.), *From Animals to Animats 4: Proceedings of Fourth Conference on Simulation of Adaptive Behavior* (Cambridge, MA: MIT Press).
- Gould S. J. (1991). Exaptations: A crucial tool for an evolutionary psychology. *Journal of Social Issues*, 3, 43-65.
- Gruau F. (1995). Automatic definition of modular neural networks. *Adaptive Behavior*, 2, 151-183.
- Harvey I., Husband I. and Cliff, D. (1994). Seeing the light: artificial evolution, real vision, in: D. Cliff, P. Husband, J-A Meyer, and S. Wilson, (eds), *From Animals to Animats 3*, Proceedings of 3rd International Conference on Simulation of Adaptive Behavior (Cambridge, Mass, MIT Press/Bradford Books).
- Higuchi T., Niwa T., Tanaka T., Iba H., De Garis H. and Furuya T. (1992). Evolving hardware with genetic learning: A first step toward building a darwin machine, in: J. A. Meyer, H. L. Roitblat, and S. W. Wilson, (eds), *From Animals to Animats 2*, Proceedings of 2nd International Conference on Simulation of Adaptive Behavior (Cambridge, Mass, MIT Press).
- Holland J. H. (1975). *Adaptation in Natural and Artificial Systems* (Ann Arbor, Mich., University of Michigan Press).
- Jacobi N., Husband P. and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics, 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) 353-367.
- Lewis M.A., Fagg A.H., Soderstrom A. (1992). Genetic programming approach to the construction of a neural network for control of a walking robot, in: *Proceedings of the IEEE International Conference on Robotics and Automation* (Nice, France).
- Mataric M. J. (1994) Learning to behave socially, in: D. Cliff, P. Husbands, J. A. Meyer and S. Wilson (eds.), *Proceedings of the Eleventh International Conference on Machine Learning* (Morgan Kaufman Publishers Inc., New Brunswick, NJ) 181-189.
- Mataric M. J., Cliff D. (1996). Challenges in evolving controllers for physical robots, in "Evolutionary Robotics", special issue of *Robotics and Autonomous Systems*, (19) 1.
- Miglino O., Nafasi K. and Taylor C. (1995). Selection for Wandering Behavior in a Small Robot. *Artificial Life*, 2, 101-116.
- 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. and Jenne 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., 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).
- Nolfi S., Florano D., Miglino O. and 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., Elman J.L. and Parisi D. (1994). Learning and Evolution in Neural Networks. *Adaptive Behavior*, 1, 5-28.
- Nolfi S. and Parisi D. (1995). Evolving non-trivial behaviors on real robots: an autonomous robot that picks up objects, in: M. Gori and G. Soda (eds), *Topics in Artificial Intelligence*, Proceedings of Fourth Congress of the Italian Association for Artificial Intelligence (Berlin: Springer Verlag).
- 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: University of Bari).
- Nolfi S. and Parisi D. (in press). Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*.
- Nolfi S. (in press). Using emergent modularity to develop control systems for mobile robots. *Adaptive Behavior*.
- Scheier C. and 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) 862-875.
- Scheier C. and Lambrinos D. (1995). Adaptive classification in autonomous agents. *Technical Report*, AILab (Computer Science Department, University of Zurich).
- Steels L. (1994). Emergent functionality in robotic agents through on-line evolution, in: R.A. Brooks and P. Maes (eds.), *Proceedings of fourth International Conference on Artificial Life* (Cambridge, Mass, MIT Press).
- Thomson A. (1995). Evolving electronic robot controllers that exploit hardware resources, 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) 353-367.
- Vanio M., Schonberg T., Halme A. and Jakubik, P. (1995). Optimising the performance of a robot society in structured environment through genetic algorithms, 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) 733-746.
- Yamauchi B. and Beer R. (1994). Integrating reactive, sequential, and learning behavior using dynamical neural networks, in: D. Cliff, P. Husband, J-A Meyer, and S. Wilson (eds.), *From Animals to Animats 3*, Proceedings of 3rd International Conference on Simulation of Adaptive Behavior (Cambridge, Mass, MIT Press/Bradford Books).