Reducing Human Design and Increasing Adaptability in Evolutionary Robotics

DARIO FLOREANO
Mantra Center for Neuro-Mimetic Systems
Swiss Federal Institute of Technology
Mantra-LAMI-INF-EPFL, CH-1015 Lausanne
Phone/Fax: +41 21 693 6696/5263
E-mail: floreano@di.epfl.ch

1 Introduction

Evolutionary Robotics is a technique for automatic creation of autonomous robots that is inspired upon the Darwinian principle of selective reproduction of the fittest. It is a new approach on its own which looks at robots as autonomous artificial organisms that develop their own skills in close interaction with the environment without human intervention. Heavily drawing from natural sciences like biology and ethology, evolutionary robotics makes use of tools like neural networks, genetic algorithms, dynamic systems, and bio-morphic engineering.

Although the term "Evolutionary Robotics" has been introduced only quite recently by Cliff, Harvey, and Husbands [3], the idea of representing the control system of a robot as an artificial chromosome subject to the laws of genetics and of natural selection dates back to the end of the 80's when the first simulated artificial organisms with a sensory-motor system began evolving on computer screens in some labs around the world. At that time, however, real robots were still machines that required accurate programming efforts and careful manipulation. Only in the last years, a few engineers began questioning some of the basic principles of robot design and came up with a new generation of robots that shared important

characteristics with simple biological systems: robustness, simplicity, small size, flexibility, modularity (e.g., see [2, 22, 15]). Above all, these robots were designed so that they could be programmed and controlled by people with different backgrounds and levels of technical skills. Progress in robot design allowed the application of evolutionary techniques to physical autonomous agents, which set the stage for a whole new range of robot programming techniques.

The basic idea behind Evolutionary Robotics goes as follows. An initial population of different artificial chromosomes, each encoding the control system (and sometimes the morphology) of a robot, are randomly created and put in the environment. Each robot (physical or simulated) is then let free to act (move, look around, manipulate) according to the the genetically specified controller while its performance to carry out a certain task is automatically evaluated. The fittest robots are then allowed to mate and reproduce; in practice, this is achieved by crossing over (swapping parts of) copies of their genetic material with small random mutations. This process is repeated for a certain number of generations until an individual is born which satisfies the performance criterion (fitness function) set by the experimenter.

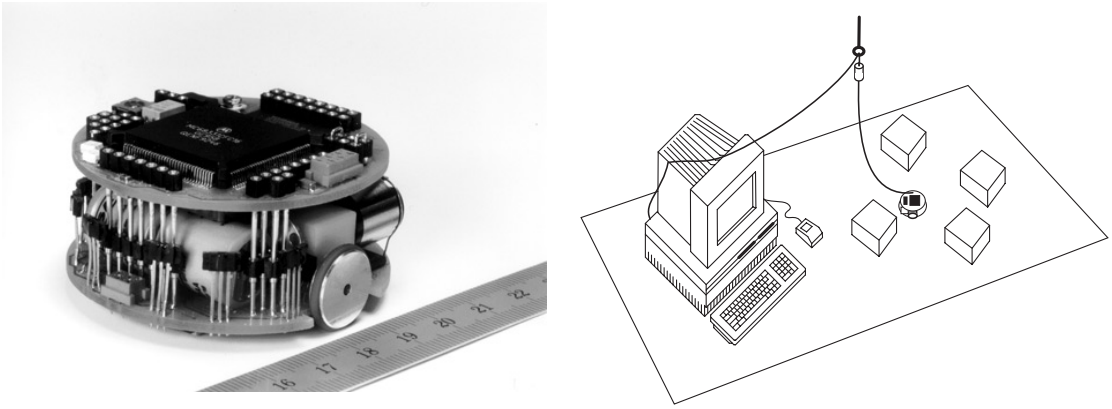
In this paper, I shall present a survey of some work done in Evolutionary Robotics by myself and in collaboration with some colleagues, mainly Francesco Mondada and Stefano Nolfi. This paper intends to be a gentle introduction to Evolutionary Robotics and, therefore, it will not include much technical detail which interested readers will find in cited articles. The presentation will be structured according to two issues which I feel are important for practical use and application of Evolutionary Robotics: *human design* and *adaptability* of evolved robots. I shall also briefly address issues such as Evolutionary Robotics in Artificial Life and the role of computer simulation. Finally, I shall conclude with some indications on future directions of research and development.

1.1 Running Evolutionary Experiments on a Single Mobile Robot

Most of the work presented here has been carried out on the miniature mobile robot Khepera initially developed by Francesco Mondada, Edo Franzi, and André Guignard at the Microcomputing Laboratory of the Swiss Fed-

eral Institute of Technology in Lausanne.

Khepera has many characteristics which make it a suitable tool for investigating evolutionary techniques in autonomous robotics. It has a circular shape (Figure 1a), with diameter of 55 mm, height of 30 mm, and weight of 70 g, and is supported by two wheels and two small Teflon balls. The wheels are controlled by two DC motors with incremental encoder



a) b)

Figure 1: a) Khepera, the miniature mobile robot. b) Evolution of neurocontrollers on a single mobile robot.

(10 pulses per mm of advancement of the robot), and can move in both directions.

Khepera is a modular system which can be extended by adding several different turrets which are simply plugged in on the top of its base. In its basic configuration, the robot is provided with eight infra-red sensors which can work either in active or passive mode. In active mode, each sensor emits infrared light and measures the quantity of reflected light. This is also called the *proximity* modality, because activation of each sensor is inversely-proportional to its distance from an object. In passive mode, each sensor simply measures the infrared component of ambient light, which gives a rough estimate of light intensity in the environment. Six sensors are positioned on one side of the robot (front), the remaining two on the other side (back). A Motorola 68331 controller with 256 Kbytes of RAM and 512 Kbytes ROM manages all the input-output routines and can communicate via a serial port with a host computer. It includes onboard rechargeable batteries which give a limited autonomy of

approximately 30 mins. However, it can also use electric power supplied by an external computer via an RS232 link. Later on, where appropriate, I shall briefly describe other modules that were used in some experiments: a laser positioning device, and a linear-array vision system.

The communication protocol of Khepera can exploit all the power and disk size available in a workstation by letting high-level control processes (genetic operators, neural network activation, variables recordings, etc.) run on the main station while low-level processes (sensor-reading, motor control, and other real time tasks) run on the on-board processor (Figure 1b). This was the solution employed in almost all our experiments. Khepera was attached via a serial port to a Sun SparcStation (as a matter of fact, the robot can be attached to any computer type) by means of a lightweight aerial cable and specially designed rotating contacts which eliminated the problem of cable twisting. In this way, while the robot was running, we could keep track of all the populations of organisms that were born, tested, and passed to the genetic operators, together with their "personal life files". At the same time, we could also take advantage of specific software designed for graphic visualization of trajectories and sensory-motor status while the robot was evolving. Starting from an initial population of randomly created binary strings, the software running on the workstation (Figure 1b) takes one string at a time, decodes it into a corresponding neural network which receives input from the robot sensors and sends its output to the motors through the serial cable every 100 ms, and computes the fitness value while the robot is running. Every individual (string) in the population is evaluated on the physical robot for some minutes and, at the end, the final fitness value is normalized by the number of actions and stored away for selective reproduction. Between any two individuals in the population, the robot is given a random sequence of actions for a few seconds in order to avoid that adjacent individuals start from a similar position (which would happen quite often during the initial generations when most of the neurocontrollers tend to get stuck against a wall). Using this method, the whole evolutionary development was carried out on a single robot without human intervention. While the robot automatically evolved in its own room, we could follow its progress from our workstation a few offices away. It should be noted, however, that the software implementing the genetic development of neural networks [8] is

very simple and could be easily optimized to fit entirely into the robot processor.

2 Reducing Human Design

Although it is possible to carry out artificial evolution on mobile robots without human intervention, human design still plays a fundamental role at early stages when the user must choose the most suitable type of controller, its genetic encoding, and the fitness function. Different controller types have been used in the last few years, including neural networks [17], modular computer programs [25], and classifier systems [6]. Although each structure has its own advantages and limits, there is not yet enough evidence for the superiority of any one type with regard to generality and behavior complexity of the evolved robot. In the work described below, I have used artificial neural networks because they have potentially interesting dynamics and are suitable for further adaptation during life of each individual. However, for what concerns the topics of this paper, the choice of neural network is rather arbitrary and the general arguments here exposed hold for any type of controller type. Regarding genetic encoding, various researchers have devised different techniques that range from the very simple "direct encoding" [32] used here to the morphogenetic specification of variable neural architectures (e.g., [24]). Whereas the latter allows for evolution of potentially more complex neural architectures, direct genetic encoding assumes a fixed architectures for all individuals in the population and evolves only its parameter values which are represented one-by-one on the genetic string (e.g., the synaptic weight values). Direct encoding strategies are suitable for small neural networks whose architecture is constrained by the structure of the robot sensory-motor system and by other considerations on system requirements (for example, the presence or absence of hidden units in perceptrons). Although there is not yet enough experimental evidence for the superiority of one method or the other with regard to behavior complexity of the evolved robots, morphogenetic approaches are likely to be a more powerful solution for future research.

Whatever controller type and encoding strategies one decides to employ, successful evolution of a desired behavior heavily depends on the fitness function. Although by now there is a "library" of fitness functions avail-

able in the literature for evolving a limited number of behaviors, these functions are suitable only for some robot morphologies and types of environment. The difficulty of designing an appropriate fitness function for an autonomous robot is mainly due to the fact that the function itself can be computed only using information available to the robot via its own sensors. In autonomous robotics, as opposed to computer simulations, it is difficult or impossible to exploit any external source of information which tells the genetic algorithm how good an individual is. To clarify this point, let us imagine to evolve a controller for a robot that must explore a surface optimally (that is, passing over the largest number of different locations in the shortest possible time). If we do this in simulation, there are several simple and straightforward fitness functions that we might use. For example, we might lay a grid over the surface and give our simulated robot a limited life time (say, 1,000 actions); then, the fitness function could simply be the number of *new* locations visited by the robot divided by its life time. When we use a real robot, it is impossible to know when it has visited a new location. Of course, we could employ a global positioning device which gives us the x, y coordinates of the robot, but then the experiment looses much of its relevance for application purposes where exploration is needed mainly for *unknown* environments.

Typically, a fitness function combines local information available from the robot sensors (including internal states of the robot, such as level of battery charge) with knowledge of environmental constraints and task requirements. A further complication comes from the fact that information available from the robot sensors is local not only in space (as for the example given above), but also in time. This means that the fitness function can appropriately select individuals only if there is immediate sensory evidence that their behavior during lifetime has accomplished at least a portion of the desired task. If the desired behavior specified in the fitness function can be accomplished only by achieving suitable performance on a sequence of sub-behaviors, it might happen that individuals at the initial generation will never receive fitness values larger than zero. A possible solution to this problem is to allocate a fitness component for each sub-behavior; however, if the resulting fitness function has several local maxima (that is, there are several sub-behaviors which generate a non-zero fitness score), the evolutionary run might settle in one of them, displaying what is known as

"premature convergence." Another solution is *incremental evolution* [16] which consists in dividing the evolutionary process in several successive stages. At each stage, which lasts for a variable number of generations, a new fitness function selects individuals that display a given sub-behavior, until the population converges to the final desired behavior.

Both the methods described above are a viable solution, although they require some knowledge on task decomposition for fitness design. However, here I shall describe another way of circumventing the problem of fitness design. Instead of describing the desired behavior in the fitness function, the environment is enriched with properties that provide several alternative ways of achieving different levels of performance. Now, it is the environment itself that puts pressure on the individuals in order to achieve more complex behaviors, whereas the fitness function, if any, is rather simple and steers evolution in the desired direction. In the following subsections, I shall review three different experiments that show how effort in fitness design can be reduced by increasing the complexity of the environment.

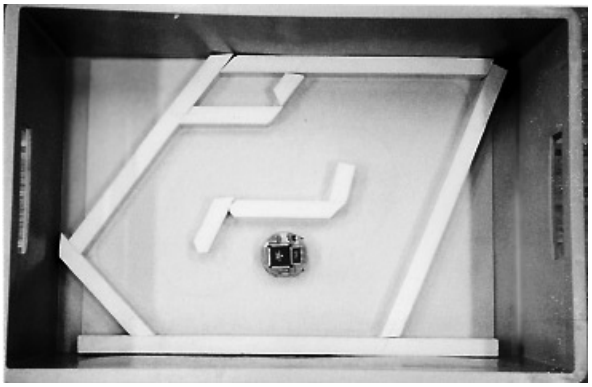
2.1 Navigation with Obstacle Avoidance

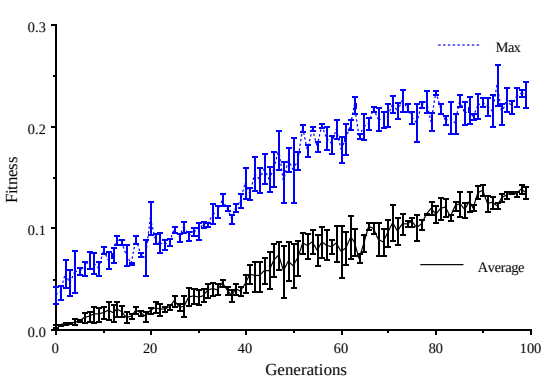
The goal of this experiment was to evolve a neurocontroller capable of performing straight navigation while avoiding obstacles [10]. The robot was put in an environment consisting in a sort of circular corridor whose external size was approx. 80x50 cm (Figure 2a). The walls were made of light-blue polystyrene and the floor was a gray thick paper. The robot could sense the walls with the IR proximity sensors. Since the corridors were rather narrow (8-12 cm), some sensors were slightly active most of the time. The environment was within a portable box positioned in a room always illuminated from above by a 60-watt bulb light.

The fitness function Φ was computed using only information available from the robot sensors

$$\Phi = V\left(1-\sqrt{\Delta v}\right)(1-i) \tag{1}$$

$$\begin{aligned} 0 &\leq V \leq 1 \\ 0 &\leq i \leq 1 \\ 0 &\leq i \leq 1 \end{aligned}$$





a) b)

Figure 2: a) Environment for the experiment on navigation and obstacle avoidance. b) Population average fitness and best individual fitness at each generation. Values are averaged over three runs (S.E. displayed).

where V is a measure of the average rotation speed of the two wheels, Δv is the algebraic difference between the signed speed values of the wheels (positive is one direction, negative the other) transformed into positive values, and i is the activation value of the proximity sensor with the highest activity. The function Φ has three components: the first one is maximized by wheel speed (without regard to direction of rotation), the second by straight direction, and the third by obstacle avoidance.

The neural network architecture was fixed and consisted of a single layer of synaptic weights from eight input units (clamped to the sensors) to two output units (directly connected to the motors) with mobile thresholds, logistic activation functions, and discrete-time recurrent connections only within the output layer. The evolutionary methodology was that described in subsection 1.1.

Khepera learned to navigate and avoid obstacles in less than 100 generations (Figure 2b), although around the 50th generation the best individuals already exhibited an almost optimal behavior. Their navigation was very smooth, they never bumped into walls and corners while trying to keep a straight trajectory. They performed complete laps of the corridor without turning back (Figure 3).

Each fitness component was necessary to develop this behavior. Without the first component, a robot standing still far from a wall would achieve maximum fitness. Without the second component, maximum fitness could

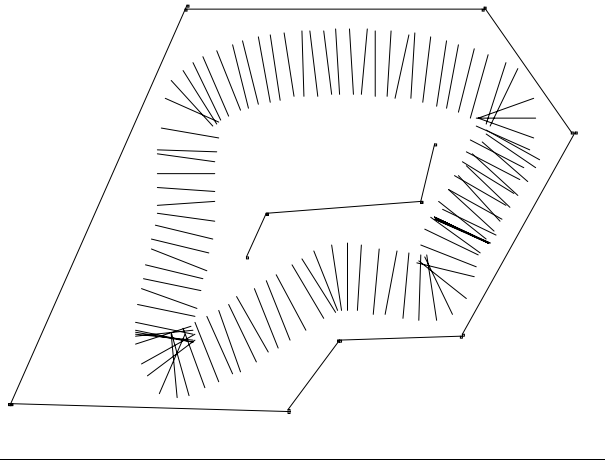


Figure 3: The trajectory performed by one of the evolved robots. Segments represent successive displacements of the axis connecting the two wheels. The direction of motion is counter-clockwise.

be easily achieved by fast spinning in the same place (wheels turning at maximum speed in opposite directions) far from walls. Finally, without the third component, evolution would develop robots moving straight (frontally or backward) at maximum speed until they crashed against a wall.

Despite the fact that the fitness function was accurately designed to evolve the desired behavior, a number of interesting aspects evolved as a side effect of the interaction with the environment. For example, despite the fact that the robot could theoretically learn to move either forward of backward (because it has a circular shape), the best individuals in all our runs developed a frontal direction of motion, corresponding to the side where there are more sensors. Robots moving backward sometimes were stuck into obstacle which could not be perceived and, since the first fitness component would give values close to zero, these individuals would not be selected for reproduction. Similarly, the best individuals developed an optimal cruising speed which was not the maximum available, but was perfectly adapted to the refreshing rate of the sensors. Robots moving faster would crash into obstacles before having detected them and would therefore soon disappear from the population. Finally, the best individuals developed appropriate recurrent connections at the output layer which worked as a tie-breaking mechanism when symmetric stimulation of sen-

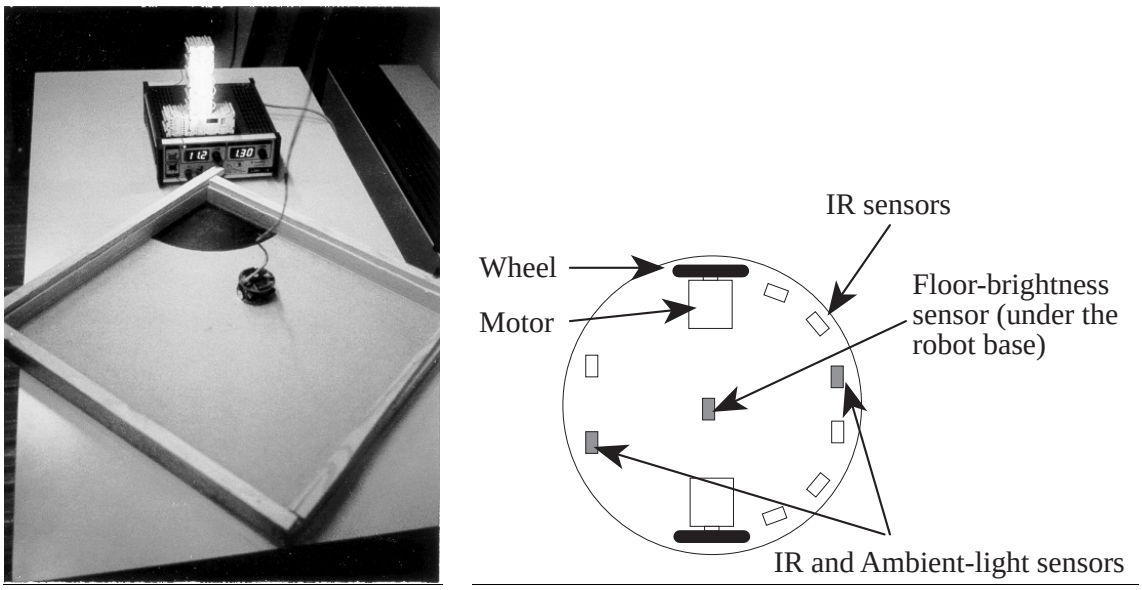
sors on both sides would generate equal and opposite signals to the wheels. Such robots would never get stuck in corners while retaining normal navigation abilities in all other situations.

2.2 Orientation and Homing

The goal of this new experiment [11] was to test the hypothesis that, when employing an evolutionary procedure, more complex behaviors do not necessarily have to be specified in the objective fitness function, but rather emerge from a mere change of the physical characteristics of the robot and of the environment described in the previous experiment. More precisely, we were interested in observing whether the robot discovered the presence of a place where it could recharge its (simulated) batteries and modify its global behavior by using an even simpler version of the fitness function employed in the previous experiment.

The environment employed for the evolutionary training consisted of a 40x45 cm arena delimited by walls of light-blue polystyrene and the floor was made of thick gray paper (Figure 4a) as in the previous experiment. A 25 cm high tower equipped with 15 small DC lamps oriented toward the arena was placed in one corner. The room did not have other light sources. Under the light tower, a circular portion of the floor at the corner was painted black. The painted sector, that represented the recharging area, had a radius of approximately 8 cm and was intended to simulate the platform of a prototype of battery charger currently under construction. When the robot happened to be over the black area, its simulated battery became instantaneously recharged.

Khepera was equipped with its basic set of eight infrared sensors (proximity sensors) whose activation is inversely proportional to the distance from an object. Two sensors, each on one side of the body, were also enabled for measuring ambient light. Additionally, another ambient light sensor was placed under the robot platform, pointing downward, and its signal was thresholded so that it was always active, except when over the black painted area in the corner (Figure 4b). The robot was provided with a simulated battery characterized by a fast linear discharge rate (max duration: approx. 20 seconds), and with a simulated sensor giving information about the battery status.



a) b)

Figure 4: a) The environment of the experiment on battery recharge. The light tower is positioned in the far corner over the recharging area which is painted black. There are no other light sources in the room. b) Sensory-motor layout of the robot.

The neural network controlling the robot was a multi-layer perceptron of continuous sigmoid units. The hidden layer consisted of 5 units with recurrent connections [7]. Each robot started its life with a fully charged battery which was discharged by a fixed amount at each time step: a fully charged battery allowed a robot to move for 50 time steps. If the robot happened to pass over the black area the battery was instantaneously recharged and, thus, its life prolonged. An upper limit of 150 steps was allowed for each individual, in order to eventually terminate the life of robots that remained on the recharging area or that regularly passed over it.

The fitness function Φ was a simpler version of that employed in the previous experiment,

$$\Phi = V(1-i) \tag{2}$$

$$\begin{aligned} 0 &\leq V \leq 1 \\ 0 &\leq i \leq 1 \end{aligned}$$

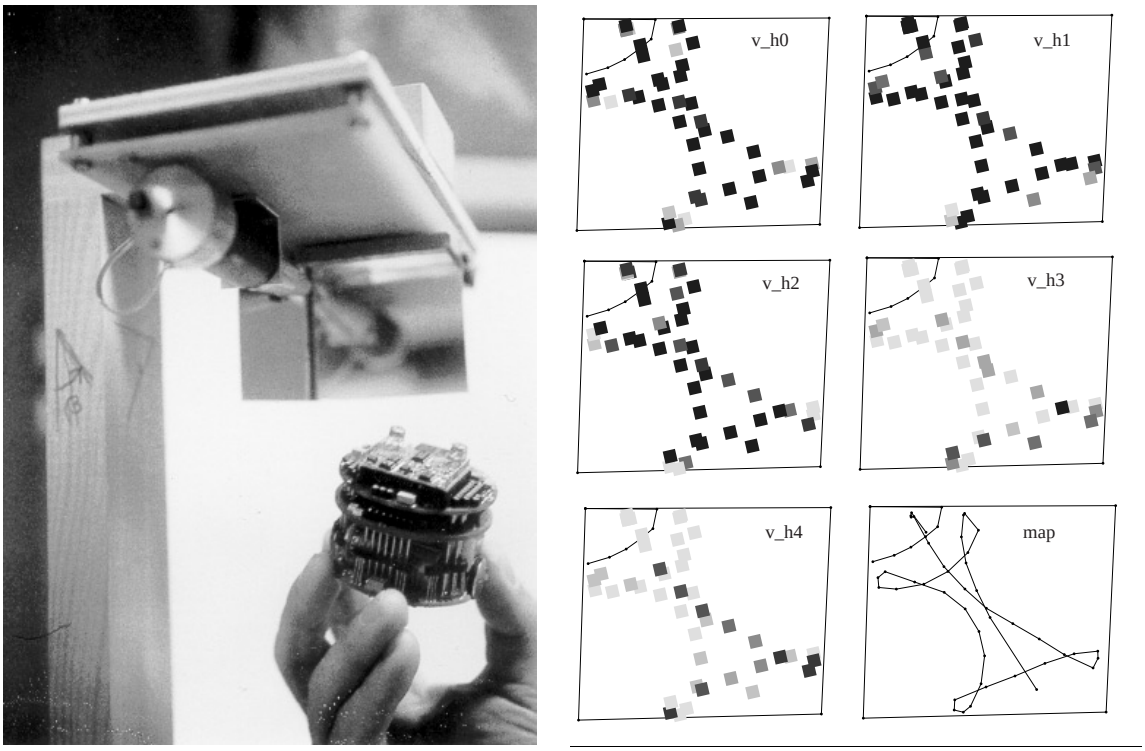
where the component responsible for straight motion had been removed:

thus, a robot could achieve a reasonable performance even by simply spinning in a place far from the walls. The fitness value was computed and accumulated at each step, except when the robot was on the black area (although later observations showed that the fitness function itself yielded values extremely close to 0 when the robot was on the black area[1]). The accumulated fitness value of each individual (which depended both on the performance of the robot and on the length of its life) was then divided by the maximum number of steps (150) and stored away for the genetic operators. It should be noted that locating and passing over the recharging area is not treated as one of the main goals that the robot should achieve, but only as a possible behavioral strategy that could emerge to exploit the characteristics of the robot and of the environment.

In order to visualize the trajectory and correlate neural activity with behavior, we installed a laser device on the top of the environment and equipped Khepera with an additional turret for detecting laser rays and calculating its own x, y position using the processor on the additional turret (Figure 5a). This information *was not passed to the controller*, but used only for analyzing behavior and neural dynamics *after evolutionary training*. The measuring device was synchronized with the network activation; therefore, for every network update, we could plot the neural activity against the robot location in the environment.

Figure 5b shows the behavior and neural activity of the best individual at generation 240. As most of the best individuals in the final generations, it navigated around the environment, avoiding the walls and, only when the battery was almost discharged, it returned to the recharging area. Once on the recharging area, the robot quickly turned on itself and resumed navigation in the arena. This behavior was maintained indefinitely or until we stopped its life. In a further set of tests [11], it was shown that the internal hidden units were specialized for different aspects of behavior. While one unit was in charge of performing reactive obstacle avoidance, another one developed a spatial representation of the environment which, combined with information on battery charge, allowed the robot to compute exactly when to initiate the homing behavior depending on its distance from the recharging area and the residual energy level.

[1]Indeed most of the time 0.0: due to the small size of the area and to the vicinity of the walls, both components are very close to zero.



a) b)

Figure 5: a) A close view of the robot with the turret for capturing laser signals and the laser device on the background. b) Visualization of the hidden node activations (five hidden nodes) while the best robot of the final generation moves in the environment. Darker squares mean higher node activation. The robot starts in the lower portion of the arena. The bottom-right window plots only the trajectory. The recharging area is visible in the top left corner.

2.3 Competitive Co-Evolution

In the previous section, I have shown how the introduction of further elements in the environment and in the robot could alleviate the effort in fitness design while generating more complex behaviors than the previous experiment. In this section, I shall describe another way of reducing fitness design by augmenting the dynamical properties of the environment. Instead of having a single robot, two robots are co-evolved in competition with each other [14].

Competitive co-evolution has recently attracted considerable interest in the community of Artificial Life and Evolutionary Computation. In the simplest scenario of two co-evolving populations, fitness progress is

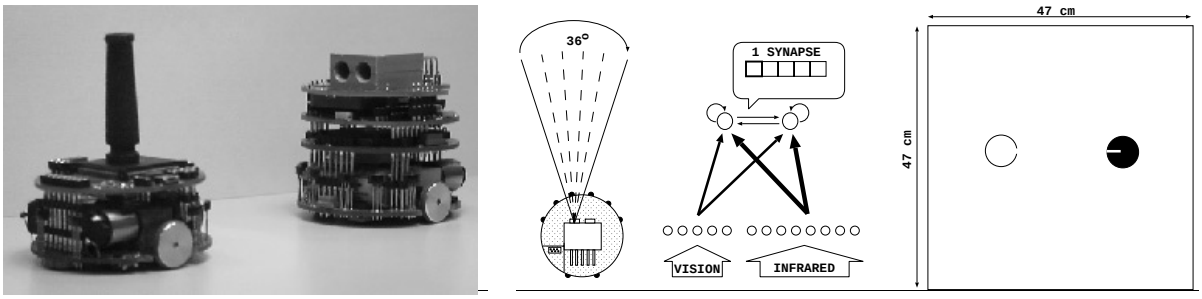
achieved at disadvantage of the other population's fitness. Although it is easy to point out several examples of such situation in nature (e.g., competition for limited food resources, host-parasite, predator-prey), it is more difficult to analyze and understand the importance and long-term effects of such "arms races" on the development of specific genetic traits and behaviors. An interesting complication is given by the "Red Queen effect"[2] whereby the fitness landscape of each population is continuously changed by the competing population. Given the relative lack of empirical evidence for the importance of the Red Queen effect on biological evolution, Artificial Life techniques seem well-suited to study this penomenon [4]. From a computational perspective, competing co-evolutionary systems are appealing because the ever-changing fitness landscape, caused by the struggle of each species to take profit of the competitors' weaknesses, could be potentially exploited to prevent stagnation in local maxima.

Cliff and Miller realised the potentiality of co-evolution of pursuit-evasion tactics in evolutionary robotics. In the first of a series of papers [21], they provided an extensive review of the literature in biology and in differential game theory and introduced their 2D simulation of simple robots with "eyes". Later, they proposed a new set of performance and genetic measures in order to describe evolutionary progress which could not be otherwise tracked down due to the Red Queen effect [4]. Recently, they described some of the results where simulated robots with evolved eye-morphologies could either evade or pursue their competitors of several generations earlier and proposed some applications of the approach in biology and in the entertainment industry [5].

Despite the promising achievements described above, if one carefully looks at the results described in the literature focusing on competitive co-evolution of pursuit-evasion behaviors, it is easy to notice that co-evolutionary benefits often come at the cost of several thousand individuals per population [26], several hundred generations [5], or repeated trials of evolutionary runs with alternating success [28]. Moreover, since all the experiments have been conducted in simulation, often the results cannot be directly applied to real robots, either because agent descriptions are too abstract or technically unfeasible, or because the fitness function takes into

[2]The Red Queen is a figure, invented by novelist Lewis Carroll, who was always running without making any advancement because the landscape was moving with her.

account global information (such as the distance between the competing agents). All these facts seem to greatly limit any prospect of exploiting the Red Queen effect for evolution of robotic controllers in the real world and for engineering purposes. The focus of the experiment here described is an investigation of the feasibility of this approach in more realistic conditions for evolutionary robotics.



a) b)

Figure 6: a) Right: The Predator is equipped with the vision module (1D-array of photoreceptors, visual angle of 36°). Left: The Prey has a black protuberance which can be detected by the predator everywhere in the environment, but its maximum speed is twice that of the predator. Both Predator and Prey are equipped with 8 infrared proximity sensors (max detection range was 3 cm in our environment). b) Left and center: Details of simulation of vision, of neural network architecture, and of genetic encoding. The prey differs from the predator in that it does not have 5 input units for vision. Each synapse in the network is coded by five bits, the first bit determining the sign of the synapse and the remaining four its strength. Right: Initial starting position for Prey (left, empty disk with small opening corresponding to frontal direction) and Predator (right, black disk with line corresponding to frontal direction) in the arena. For each competition, the initial orientation is random.

The main goal of the experiments described here consists in studying the feasibility of co-evolutionary pursuit-evasion for evolving useful neurocontrollers for two Khepera robots in a simple but realistic scenario. We decided to study pursuit-evasion as a metaphor for predator-prey, this being a quite common and suggestive situation in nature. As often happens, predators and preys belong to different species with different sensory and motor characteristics. Thus, we employed two Khepera robots, one of which (the *Predator*) was equipped with a vision module while the other (the *Prey*) had a maximum available speed set to twice that of the predator (Figure 6a). Both individuals were also provided with eight infrared

proximity sensors (six on the front side and two on the back) which had a maximum detection range of 3 cm in our environment. The two species would evolve in a square arena of size 47 x 47 cm with high white walls so that the predator could always see the prey (if within the visual angle) as a black spot on a white background.

Running co-evolutionary experiments with two or more robots within the same environment causes problems with the cables that connect the robots to the workstation for power supply and information exchange. Therefore, we decided to resort to a particular type of simulation extensively tested on Khepera which consists in sampling sensor activity at different distances and angles of the robot from the objects of the world (see [20] for details). We have sampled infrared sensor activity of each robot in front of a wall and in front of another robot. These values were then separately stored away and accessed through a look-up table depending on the faced object. Simulation of the visual input required different considerations. The vision module K213 of Khepera is an additional turret which can be plugged-in directly on top of the basic platform. It consists of a 1D-array of 64 photoreceptors which provide a linear image composed of 64 pixels of 256 gray-levels each, subtending a view-angle of 36°. The optics are designed to bring into focus objects situated at distances between 5cm and 50cm while an additional sensor of light intensity automatically adapts the scanning speed of the chip to keep the image stable and exploit at best the sensitivity of receptors under a large variety of illumination intensities. The K213 vision turret incorporates a private 68HC11 processor which is used for optional low-level processing of the scanned image before passing it to the robot controller. One of these options is detection of the position in the image corresponding to the pixel with minimal intensity (in this case, only one byte of information is transmitted). Therefore, instead of simulating the response of the 1D-array of receptors resorting to complex and time-consuming ray-tracing techniques, we exploited the built-in facility for position detection of the pixel with minimal intensity and divided the visual angle in five sectors corresponding to five simulated photoreceptors (Figure 6b). If the pixel with minimal intensity was within the first sector, then the first simulated photoreceptor would become active; if the pixel was within the second sector, then the second photoreceptor would become active; etc. We made sure in a set of preliminary measurements

that this type of input reduction was largely sufficient to reliably capture and represent all the relevant visual information available to the predator.

The robot controllers had the same architecture used for the experiment described in section 2, but the predator had 5 more input units corresponding to the visual module. Two populations of 100 individuals each were co-evolved for 100 generations. Each individual was tested against the best competitors of the ten previous generations (a similar procedure was used in [28, 26, 4]) in order to improve co-evolutionary stability. For each competition, the prey and predator were always positioned on a horizontal line in the middle of the environment at a distance corresponding to half the environment width (Figure 6b), but always at a new random orientation. The competition ended either when the predator touched the prey or after 500 motor updates (corresponding to 50 seconds at maximum on the physical robot). The fitness function Φ_c for each competition c did not require any sensor or motor measurement, nor any global position measure; it was simply *TimetoContact* normalized by the maximum number of motor updates TtC for the predator pr, and $1 - TtC$ for the prey py, further averaged over the number of competitions. Therefore the fitness values were always between 0 and 1, where 0 means worst.

Six evolutionary runs were performed, each lasting 100 generations. In all cases, after a few generations both the predator and the prey increased their fitness value and, for the remaining generations, we observed a set of oscillations in counterphase where either the predator or the prey reported better performance over the competitor. When we looked at the behavior of the two robots, we observed spontaneous evolution of obstacle avoidance, visual tracking, object discrimination (prey vs. wall), following, and a variety of other temporary behaviors (that is, lasting only few generations) which were tuned to the competitor behavior.

However, the tight evolutionary dynamics between the two competitors implied that the individuals of the last generation were not necessarily the best individuals of all generations, as it is usually the case in single-agent evolution. Rather, at each generation the best individuals are those individuals that report the best performance against the best competitors of the previous ten generations. The simplest way to discover the best predators and preys, is to organize a Master Tournament where each best individual is tested ten times against each best competitor of all genera-

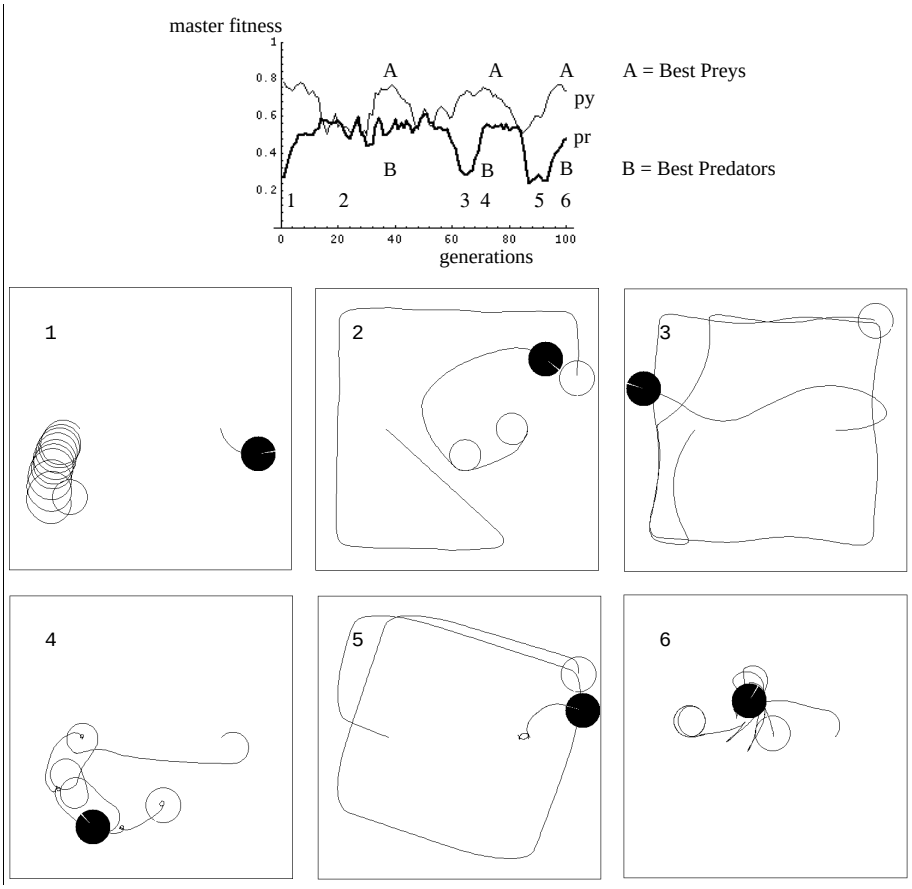


Figure 7: Top: Fitness of best individuals in Master Tournament. Letters indicate position of best preys and best predators. Numbers indicate position of individuals whose competitions are displayed below. Bottom: Behaviors recorded at interesting points of co-evolution, representing typical strategies. Black disk is predator, white is the prey.

tions. Top of Figure 7 shows the master fitness for each best individual across generations (fitness values are averaged over ten competitions and over hundred tournaments). A Master Tournament tells us two things: At which generation we can find the best prey and the best predator, and at which generation we are guaranteed to observe the most interesting tournaments. The first aspect is important for optimization purposes and applications, the latter for pure entertainment. The best individuals are those reporting the highest fitness when also the competitor reports the highest fitness (marked by letters A and B in the graph). Instead, the most entertaining tournaments are those that take place between individuals that report the same fitness level, because these are the situations where both species have the same level of ability to win over the competitor.

In the lower part of Figure 7, behaviors of best competitors at critical stages of co-evolution, as indicated by Master Tournament data, give a more intuitive idea of how pursuit-evasion strategies are co-evolved. Initially, the predator tends to stop in front of walls while the prey moves in circles (box 1). Later, the prey moves fast at straight trajectories avoiding walls while the predator tracks it from the center and quickly attacks when the prey is closer (box 2). Interestingly, predators develop the ability to know how distant the preys are by using information on how fast their target moves in the visual field. Decrement of predator performance around generation 65 is due to a temporary loss of the ability to discriminate between walls and preys. As shown in box 3, the predator intercepts the prey, but it misses it crashing against the wall. Around generation 75, we have a typical example of the best prey (box 4); it moves in circles and, when the predator gets closer, it rapidly avoids it. This is quite interesting. Indeed, preys that move too fast around the environment sometimes cannot avoid an approaching predator because they detect it too late (IR sensors have lower sensitivity for a small cylindrical object, like another robot, than for a white flat wall). Therefore, it pays off to wait for the slower predator and accurately avoid it. However, some predators become smart enough to perform a small circle once they have missed the target, and re-attack until, by chance, the prey displays a side without IR sensors. As soon as the preys begin again moving around the environment, the predator develops a "spider strategy" (box 5): it slowly backs until it finds a wall where it waits for the fast-approaching prey. However, this strategy does not pay off when the preys stay in the same place. Finally, at generation 99 we have a new interesting strategy (box 6): the predator quickly tracks and reaches the prey which quietly rotates in small circles. As soon as the prey senses the predator, it backs and then approaches the predator (without touching it) on the side where it cannot be seen; consequently, the predator quickly turns in the attempt to visualize the prey which rotates around it, producing an entertaining dance.

Although before conducing experiments in real time on more complex robots one should devise a solution for power supply (which is one of our current efforts), these experiments have shown that competitive co-evolution is a promising technique for automatic evolution of complex behaviors without much effort in fitness design.

3 Evolution of Adaptive Agents

Artificial evolution generates controllers which are well matched to the training environment. However, in practical applications some characteristics in the environment might change unexpectedly. If genetic algorithms are the only adaptation engine employed, any change in the environment would require further evolutionary training in the new conditions. If the population of genetic strings still has sufficient diversity (that is, chromosomes are not all equal), this might be achieved faster than retraining the whole system from scratch. For example, consider the experiment on navigation and homing for battery recharge described in subsection 2.2. The robot was evolved for 240 generations in an environment with a light source on top of the recharging area (Figure 4b).

After generation 240, we positioned the light source in a different corner of the environment and continued evolutionary training [9] in the new condition. Figure 8 shows fitness values for evolutionary training on three different light positions, each time starting from the same population of generation 240. As one can see on the first column, the fitness of the best individuals returns to previous values in approximately 20 generations (the population average fitness taking longer). A further indication that the best individuals are capable to correctly locate the recharging area in the new conditions is given by the graphs of the second column which plot the number of actions which the best individual of each generation can perform.[3]

Despite these results, running evolutionary experiments on a single robot can be a time consuming process. For example, in the experiments described in subsections 2.1 and 2.2 each generation lasted approximately 40 minutes. Although evolutionary time can be considerably reduced by optimizing certain parameters of the genetic algorithm [27], it would still be desirable for practical purposes to evolve more adaptive systems. In other words, the time required for evolutionary training could be more cost-effective if the evolved system were capable of additional self-organization if and when necessary. In this way, one could evolve a controller for a certain task in a prototypical environment and then put the robot in the

[3]Let us recall that a full battery lasts 50 actions and that after 150 actions the individual is "killed" to leave place for the next one.

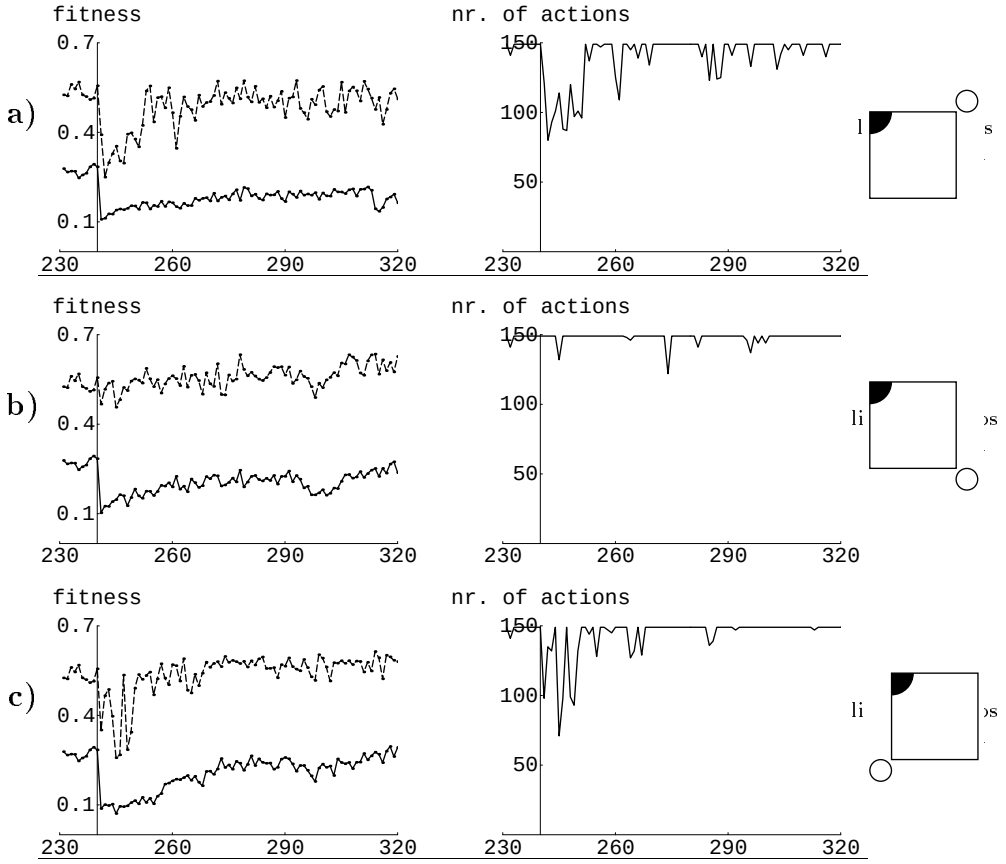


Figure 8: Re-adaptation in environments with a new light position. Each row (a, b, and c) plots –respectively– the average population fitness (continuous line) and the fitness of the best individual (dotted line) across generations, the number of actions during life for the best individual at each generation, and a sketch of the light position (small circle) in the environment (the black sector represents the charging area). For sake of comparison, each plot includes data for the last ten generations of the original run; therefore, data on re-training start in correspondence of the origin of the y-axis.

operating environment where it would fine tune its behavior to accomodate minor changes if necessary.

In nature there are at least two sources of adaptation: changes that take place on the phylogenetic scale and changes that take place during the life of each individual. To what extent these two sources of adaptation interact is still an open issue, but the role of adaptive individuals in evolving populations has interesting aspects also for our computational models (the reader is referred to [1] for a good treatment of this subject in biology and in computer science). For example, phylogenetic evolution cannot take into account temporary environmental modifications that happen during

the life of an organism; these modifications are assimilated by ontogenetic mechanisms, such as maturation and learning.

Also in evolutionary robotics it would be desirable if evolved controllers could incorporate some form of adaptation which remains operative after evolutionary training. The idea is that while artificial evolution shapes high-level properties of the controller architecture and dynamics, another form of fast adaptation tunes low-level parameters of the controller to the characteristics of the environment in which the robot is expected to operate. In the next section, I will describe some initial explorations in this direction.

3.1 Evolving Modifiable Neural Networks

Adaptation in artificial neural networks usually takes place by changing the value of the synaptic strengths [18]. In biological nervous systems, these changes are regulated by some form of Hebbian plasticity, that is synaptic modifications depend on some type of correlation between the activity of presynaptic and postsynaptic neurons.

In this experiment we explored the idea of evolving the type of Hebbian rule employed by each synapse, rather than evolving the value of the synapse itself [12]. Thus, a genetic algorithm was used to evolve neural structures that could be continuously modified during life of the individual according to the mechanisms specified in the genotype. The experimental setup and the fitnes function employed in this experiment was identical to that already described in subsection 2.1. The genotype of each individual encoded a set of parameters describing synapse properties and four Hebbian rules. Every time a phenotype was created, its synapses were intialized to small random values and could change their strength during life using one of the four rules, as specified in the genes; final strengths were not coded back into the chromosome. Thus, each decoded neural network changed its own synaptic strength configuration according to the genetic instructions and without external supervision while the robot interacted with its own environment.

Synapses were individually coded on the chromosome. Each synapse was described by a set of four properties: whether it was driving or modulatory (1 bit), whether it was excitatory or inhibitory (1 bit), its Hebbian

rule (2 bits), and its learning rate (2 bits). Each individual synapse could change its strength according to one of four basic Hebbian learning rules: pure Hebbian, postsynaptic, presynaptic, and covariance (see [30]). These rules were such that synaptic strength could not grow indefinitely, but was intrinsically bound in the range $[0.0, 1.0]$ by means of a self-limiting mechanism which depended on the current synaptic strength; this solution had the property of keeping the sign of the synapse unchanged, thus reducing the degrees of freedom of the network and putting more emphasis on the genetically evolved configuration of excitation and inhibition. When a neurocontroller was decoded from the corresponding genotype, the input units were attached to the sensors and the output units to the motors of the robot; then, each synaptic weight value w^t was initialized to small random values in the range $[0.0, 0.1]$ and updated every 300 ms according to the following discrete-time equation

$$w^t = w^{t-1} + \eta \Delta w^t \tag{3}$$

where η is the learning rate, which can assume one of four values (0.0, 0.3, 0.7, 1.0) and Δw^t is the change at time$t$ computed using one of the four Hebbian rules. If the learning rate for a given synapse was 0.0, that synapse would not change its strength during the life of the individual.

Three different runs of this experiment were made. In all cases the best individual fitness reached a maximum value around the 50th generation ($\Phi = 0.23, \pm 0.09$). All the best neural networks of the last generation could control the robot in order to keep a straight trajectory while avoiding obstacles. The evolved behaviors resulted in smooth paths around the arena (Figure 9). This ability was developed by each individual neurocontroller during the first few sensory-motor loops, whatever the inital random values assigned to the synapses. This can be seen on the left plot of figure 9 where the robot starts its adaptation in the lower portion of the environment. As time passed, the robot exhibited smoother trajectories (for example turning on sharp bends as compared to the sequence stop-back-turn-restart employed earlier on). In all the three runs the best individuals of the last generation moved in the direction where more IR sensors were placed, which provided a better sampling of the obstacles facing the robot, as in the experiment described in subsection 2.1.

By analyzing the neurocontroller dynamics while the robot behaved in

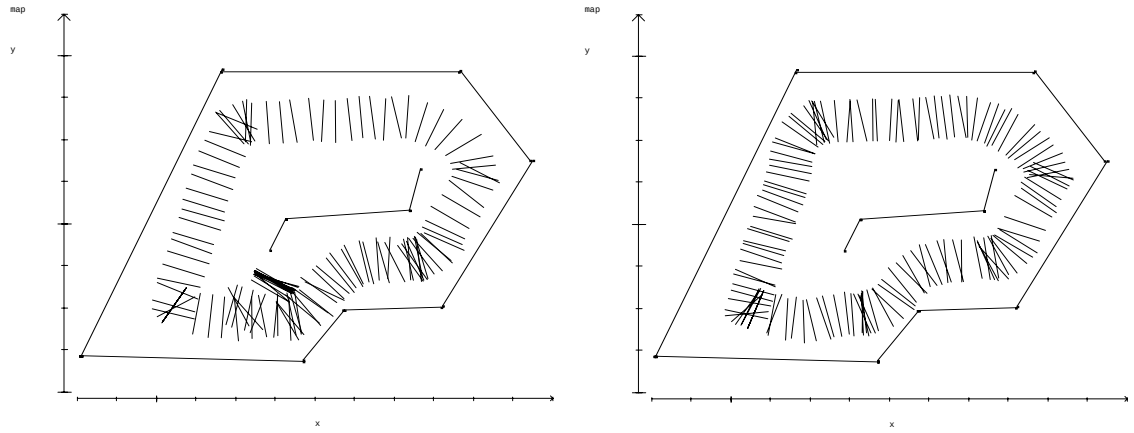


Figure 9: Trajectory of the robot that learns to navigate during life. Position data, visualized as bars representing the axis connecting the two wheels, were acquired with the laser positioning system every 100 ms. Data refer to the best individual of the last generation of one evolutionary run. Left: trajectory during the first lap (the robot starts in the lower portion of the environment and turns anti-clockwise). Right: trajectory at the second lap.

the environment, we realized that the stable behavior acquired during life was regulated by *continuously changing synapses* which were *dynamically stable*. In the conventional view, synapses are relatively stable and slow components of the nervous system. Synaptic changes are identified with the learning of new skills or acquisition of new knowledge, while neuron activations are identified with the expression of behavior and of existing knowledge.[4] Typically, acquisition of a stable behavior in a static environment corresponds to stability (no further change) of individual synapses (e.g., see [18]). Such requirement is explicitly included into the objectives (least-mean-square error minimization, energy reduction, maximization of node mutual information, etc.) from which –both supervised and unsupervised– conventional learning algorithms are derived, but it was not included into the fitness function employed here, which is defined only in behavioral terms. The functioning of our system offers a complementary –but not necessarily alternative– explanation to adaptation: Synapses are responsible for both learning and behavior regulation. Knowledge in the network is not expressed by a final stable state of the synaptic configura-

[4]This view has been recently challenged by Yamauchi and Beer [31], who have evolved and analyzed continuous-time recurrent neural networks that give the external appearance of performing reinforcement learning while, in fact, these networks have fixed connection weights and use only internal node dynamics.

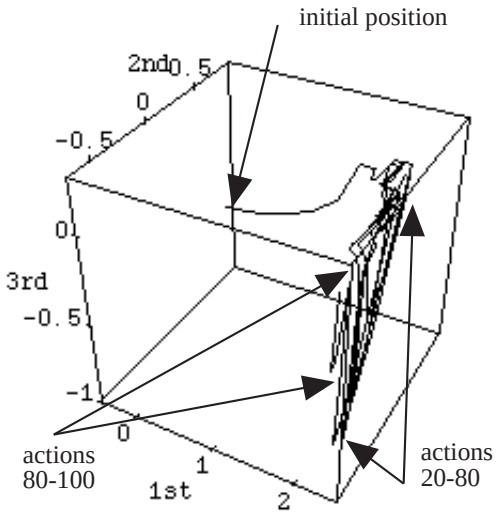


Figure 10: State-space representation of synapse dynamics during the first 100 actions (visualized on the left side of Figure 9) plotted as trajectory within the space of the first three principal components. Arrows indicate the starting position and the range of oscillation between action sequences 20-80 and 80-100. Oscillations within the subspace of the third (smallest) component correspond to trajectory adjustments.

tion, but rather by a dynamic equilibrium point in a n-dimensional state-space (where n is the number of synapses). Figure 10 plots the trajectory of synaptic change in the reduced state-space of the first three principal components of the recorded synaptic vectors during the first 100 actions of the individual displayed in Figure reffig:traj.glearn.eps. During the first 6 action the systems moves toward a subregion of the space for which there is no change in the first two principal components; residual variation along the space plane corresponding to the third principal component corresponds to trajectory adjustments and is further reduced as the robot gradually tunes its path to the geometry of the environment.

The adaptation abilities of the neurocontroller were limited only to those variations that were encountered in the training environment (e.g., walls with different curvatures). To this extent, such a plastic system evolved in stationary environments did not offer significant adaptation advantages w.r.t. systems with fixed synaptic weights, but this experiment showed the viability of such approach. Further analyses which are currently being carried out indicate that such plastic neurocontroller is much more resistent to sensor damages than a static controller.

3.2 Adaptive Behavior in Co-Evolutionary Systems

In order to test in dynamical environments the approach described in the previous section, we have revisited the competitive co-evolutionary scenario outlined in subsection 2.3. Both predator and prey are now co-evolved using the genetic structure described in the previous subsection –thus adapting synaptic weights during their life– and the results are compared with those obtained in subsection 2.3. In order to make better comparisons, both the neural network architecture and the genotype length were the same in both experiments. Furthermore, another set of experiments was run where, instead of Hebbian modification, the synapses were changed by adding random noise (more details can be found in [13]). For sake of simplicity, let us label the evolution of static controllers condition 1 (i.e., the experiment described in subsection 2.3), the evolution of random synapses condition 2, and the evolution of adaptive synapses condition 3. For each condition, 6 different evolutionary runs were performed, each starting with a different seed for initializing the computer random functions. A set of pairwise two-tail t-tests of the average fitness and best fitness along generations among all the six runs, performed to check whether different seeds significantly affected the experimental outcomes, gave negative results at significance level 0.05.

A *relational measure* of performance gives us interesting information on the coupled dynamics of a co-evolved system: for example, one can derive an index of *relative performance* r_i^c by counting how often one species reports higher fitness than the competitor species at each generation for each separate run i in a specific condition c. In our setup, such index was in the range $[-100, 100]$, where -100 means that the preys always outperformed the predators, 0 means that both species were equally better or worse than the competitors, and 100 means that the predators always outperformed the preys. In condition 1 ($c = 1$) described in subsection 2.3, the average value over different runs was $\overline{R^1} = 16.67$ with standard deviation of the sample mean $\sigma = 38$, indicating that both species reported similar performances. The condition with evolutionary adaptive noise ($c = 2$) displayed an average relative performance $\overline{R^2} = 11.66$ with standard deviation of the sample mean $\sigma = 32.5$ which was not statistically different from that of the previous condition (probability value was 0.83 for t-test of the difference of

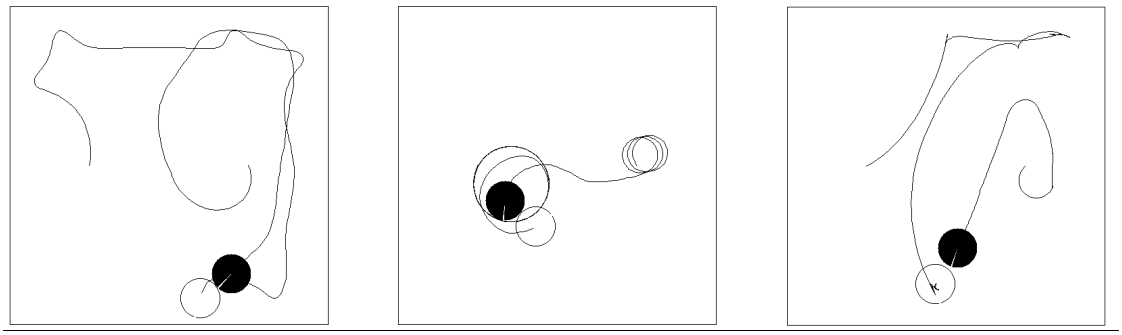


Figure 11: Behaviors of species with adaptive synapses. Black disk is predator, white is prey. **Left**. Generation 20. **Center**. Generation 70. **Right**. Generation 95.

the means between the two conditions, i.e. much bigger than significance level 0.05 typically used for rejecting the equality hypothesis). Instead, relative performance of the two species in condition 3 significantly differed from condition 1 and condition 2, $\overline{R^3} = 72$ with standard deviation of the sample mean $\sigma = 15.39$, $p < 0.01$ for a two-tailed t-test of the difference of the means. In all the six evolutionary runs predators reported higher average and best fitness values than preys. Furthermore, in all runs, the average fitness of the predator population was more stable than that of the preys.

As compared to condition 1, where the predator tended to efficiently track in only one direction, here it can turn in both directions at equal speed. In condition 1 proper tracking in both directions required accurate settings of all the visual synaptic strengths. Here, instead, since synapses are temporarily increased depending on active units [12], individual adjustments of synapses take place when and where required depending on current sensory input. The trajectory in the center image of figure 11 shows an example of synaptic adjustment. Here, while the prey rotates always around the same circle, the predator performs three turns during which synaptic values from the visual units are gradually increased; at the fourth turn, the synaptic values will be sufficiently high to cause a straight pursuit (eventually, the prey will try to avoid the predator without success). If activity-dependent synaptic change are exploited by the far-sighted predator, not the same happens for the prey. Although here preys are faster than in condition 1 and 2, especially when turning near walls (where IR sensors become active and synapses temporarily strengthen), they cannot

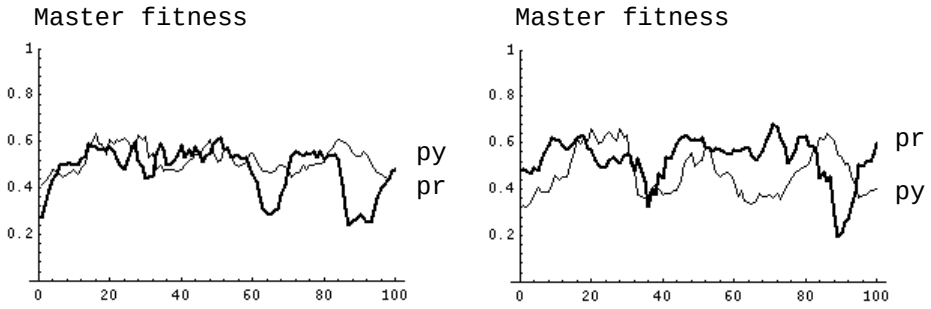


Figure 12: Master Tournament between species evolved in condition 1 (left) and Master Tournament between predator evolved in condition 3 and prey evolved in condition 1 (right).

increase their behavioral repertoire with respect to condition 1. Not even can they improve it because volatile changes of the synaptic values imply that most of the time they must re-develop on-the-fly appropriate synaptic values; although this can be well-suited for avoidance of static obstacles, it does not score better when facing another moving object.

In order to check whether such improvements were due to a real advantage of the predator, rather than to some difficulties of the preys to cope with directional-change controllers, we performed two Master Tournaments. The graph on the left of figure 12 shows the Master fitness for predators and preys co-evolved in condition 1, giving a relative performance $\overline{R} = -12$ (relative performance for the average fitness data of this run was $\overline{r_1^1} = -4$). The graph on the right of figure 12 instead shows the Master fitness for predators evolved in condition 3 against preys evolved in condition 1, giving a relative performance $\overline{R} = 42$ (relative performance for the average fitness data of this run was $\overline{r_1^3} = 50$). Had the advantage reported by predators in condition 3 been caused by underdeveloped preys rather than better predators, the Master Tournament between species evolved in different conditions should have generated opposite results.

4 Some Future Directions

In this chapter I have considered two issues which I feel important for practical applications of the Evolutionary Robotics approach: reduction of human design in the fitness function and evolution of adaptive systems.

Although my suggestions have only an empirical character, the results obtained so far seem promising.

There are several other ways of improving utility and usability of the evolutionary approach in robotics. One of these is an appropriate use of simulation as an integrated tool to develop robotic controllers. Given the long duration of experiments in evolutionary robotics, simulations can considerably speed up initial search of optimal architectures, evolutionary parameters, and fitness functions. Simulations can also be used to carry out most of evolutionary training before moving to the real robot. Initial generations run in simulation would develop the rough structure of the controller with respect to the task requirements and the environment characteristics, whereas the final generations on the real robot would fine tune the controller to the specific mechanical and physical constraints of the real-world task [23]. This type of incremental evolution (which does not necessarily imply changes in the fitness function) could later be continued on another robotic platform, such as a robot eith a different geometry or a different sensory system. At the Evolutionary Robotics workshop in 1996, Francesco Mondada showed that continuing evolution on a different robot is not only feasible, but also faster than retraining the controller from scratch on the new robot. In his experiment he continued the experiment on navigation and obstacle avoidance with the Khepera described in section 2.1 on a bigger robot, the Koala. He showed that the neurocontrollers adapted in few generations their internal paramaters to the new mechanical and geometrical characteristics of the bigger robot.

But, why not evolving controllers directly on the target robot? The reason is that if a robot has a complex geometry or poor sensors (with respect to the task requirements), then an appropriate controller for such robot would represent a very small zone in the huge space of possible controllers. In very simple words, it might be difficult for a genetic algorithm with a small population size to discover this zone. Instead, by following the *simulation-simple robot-target robot* strategy, the genetic algorithm would search a solution space which is initially relatively simple (that is, there are several zones that would correspond to viable –even if not optimal– controllers) and then progressively grows in complexity. This line of reasoning is similar to the case where one tries to evolve a complex behavior by progressively changing the fitness function [16], as described in section 2.

An incremental approach is feasible only when there is sufficient variablility in the population of genetic strings. There are no ways of telling what is the minimum necessary variability for practical applications because this depends on a number of factors, such as the type of genetic encoding employed, the genetic operators, the nature of the task, how noisy the fitness evaluation is, etc. However, there are several ways for ensuring such variability, such as injection of random strings or evolution of isolated genetic populations with occasional exchange of individuals. The stochastic process of selective reproduction together with the recombination operator would then appropriately exploit these sources of variation.

Another way of improving evolutionary development of controllers might be *Genetic Recycling*. Instead of starting an evolutionary run from randomly initialized genetic strings, one might include in the initial population some genetic strings evolved in a previous experiment. The idea is that, if the task requirements of the two experiments share some common ability (e.g., obstacle avoidance or visual tracking), previously evolved "building blocks" could be appropriately exploited for the new evolutionary run. Even when the two experiments make use of different robotic platforms or employ different environments or different fitness functions, recycling could still be beneficial. For example, some recycled strings could report non-zero performance values already at the initial generation and be selected for reproduction, recombination, and mutation, thus speeding up initial search for viable controllers. In the case where initial performance of recycled strings is not better than that of randomly initialized strings, they could still be stochastically selected by the reproduction operator and maintained in the population. The building blocks of recycled strings could then become useful at some later stage when appropriately recombined with those of other evolving strings.

Evolutionary Robotics is a young approach still open to several exciting research developments and applications. As for any novel method, there is still space for improvement, systematization, and formalization. However, successful results on real robots presented in recent years already show the practical feasibility of the method. One of the nice features of the evolutionary approach is its generality, that is the possibility to apply it to several different aspects robotics (e.g., mechanical design [19] or circuit design [29]) and to several different control specifications. Furthermore, it

can be fruitfully combined with other forms of adaptation (such as learning) and/or with traditionally pre-designed solutions.

References

[1] R. K Belew and M. Mitchell, editors. *Adaptive Individuals in Evolving Populations. Models and Algorithms.* Addison-Wesley, Redwood City, CA, 1996.

[2] R. A. Brooks. New approaches to robotics. *Science*, 253:1227–1232, 1991.

[3] D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior*, 2:73–110, 1993.

[4] D. Cliff and G. F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life*, pages 200–218. Springer Verlag, Berlin, 1995.

[5] D. Cliff and G. F. Miller. Co-evolution of Pursuit and Evasion II: Simulation Methods and Results. In P. Maes, M. Mataric, J-A. Meyer, J. Pollack, H. Roitblat, and S. Wilson, editors, *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior.* MIT Press-Bradford Books, Cambridge, MA, 1996.

[6] M. Dorigo and U. Schnepf. Genetic-based machine learning and behavior based robotics: a new synthesis. *IEEE Transactions on Systems, Man and Cybernetics*, 23:141–154, 1993.

[7] J. L. Elman. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

[8] D. Floreano. Robogen: A software package for evolutionary control systems. Release 1.1. Technical report LabTeCo No. 93-01, Cognitive Technology Laboratory, AREA Science Park, Trieste, Italy, 1993.

[9] D. Floreano. Evolutionary re-adaptation of neurocontrollers in changing environments. In P. Sincak, editor, *Proceedings of the Conference Intelligent Technologies*, volume II. Efa Press, Kosice, Slovakia, 1996.

[10] D. Floreano and F. Mondada. Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. In D. Cliff, P. Husbands, J. Meyer, and S. W. Wilson, editors, *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1994.

[11] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:396–407, 1996.

[12] D. Floreano and F. Mondada. Evolution of plastic neurocontrollers for situated agents. In P. Maes, M. Mataric, J-A. Meyer, J. Pollack, H. Roitblat, and S. Wilson, editors, *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1996.

[13] D. Floreano and S. Nolfi. Adaptive behavior in competing co-evolving species. Mantra technical report, LAMI, Swiss Federal Institute of Technology, Lausanne, 1997. Also submitted to ECAL97.

[14] D. Floreano and S. Nolfi. God save the red queen! competition in co-evolutionary robotics. In J. Koza, K. Deb, M. Dorigo, D. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Proceedings of the 2nd International Conference on Genetic Programming*, Stanford University, 1997.

[15] T. Gomi. Non-cartesian robotics. *Robotics and Autonomous Systems*, 18:169–184, 1996.

[16] I. Harvey, P. Husbands, and D. Cliff. Seeing The Light: Artificial Evolution, Real Vision. In D. Cliff, P. Husbands, J. Meyer, and S. W. Wilson, editors, *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1994.

[17] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: The sussex approach. *Robotics and Autonomous Systems*, page In press, 1997.

[18] S. Haykin. *Neural Networks. A Comprehensive Foundation.* MacMillan, Englewood Cliffs, NJ, 1994.

[19] H. H. Lund, J. Hallam, and W-P. Lee. Evolving robot morphology. In *Proceedings of the IEEE 4th International Conference on Evolutionary Computation.* IEEE Press, 1997.

[20] O. Miglino, H. H. Lund, and S. Nolfi. Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, 2:417–434, 1996.

[21] G. F. Miller and D. Cliff. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In D. Cliff, P. Husbands, J. Meyer, and S. W. Wilson, editors, *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior.* MIT Press-Bradford Books, Cambridge, MA, 1994.

[22] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. In *Proceedings of the Third International Symposium on Experimental Robotics*, Kyoto, Japan, 1993.

[23] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. Brooks and P. Maes, editors, *Proceedings of the Fourth Workshop on Artificial Life*, pages 190–197, Boston, MA, 1994. MIT Press.

[24] S. Nolfi and D. Parisi. Genotypes for neural networks. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks.* MIT Press, Cambridge, MA, 1995.

[25] P. Nordin and W. Banzhaf. An online method to evolve behavior and to control a miniature robot in real time with genetic programming. *Adaptive Behavior*, page In press, 1997.

[26] C. W. Reynolds. Competition, Coevolution and the Game of Tag. In R. Brooks and P. Maes, editors, *Proceedings of the Fourth Workshop on Artificial Life*, pages 59–69, Boston, MA, 1994. MIT Press.

[27] R. Salomon. Increasing adaptivity through evolution strategies. In P. Maes, M. Mataric, J-A. Meyer, J. Pollack, H. Roitblat, and S. Wilson, editors, *From Animals to Animats IV: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1996.

[28] K. Sims. Evolving 3D Morphology and Behavior by Competition. In R. Brooks and P. Maes, editors, *Proceedings of the Fourth Workshop on Artificial Life*, pages 28–39, Boston, MA, 1994. MIT Press.

[29] A. Thompson, I. Harvey, and P. Husbands. Unconstrained evolution and hard consequences. In E. Sanchez and M. Tomassini, editors, *Towards Evolvable Hardware. The Evolutionary Engineering Approach*. Springer Verlag, Berlin, 1996.

[30] D. Willshaw and P. Dayan. Optimal plasticity from matrix memories: What goes up must come down. *Neural Computation*, 2:85–93, 1990.

[31] B. Yamauchi and R. D. Beer. Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2:219–246, 1995.

[32] X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 4:203–222, 1993.