Bio-Inspired Computing Systems

Toward Novel Computational Architectures

Daniel Mange and Marco Tomassini (Eds.)

Contents

Chapter 1

Evolution and Learning in Autonomous Robotic Agents

Dario Floreano and Joseba Urzelai

1.1 Introduction

Evolution and learning are two forms of biological adaptation that differ in *space*, *time*, and *substrate*. Evolution is a process of selective reproduction and substitution based on the existence of a geographically-distributed population of individuals displaying some variability. Learning , instead, is a set of modifications taking place within each single individual during its own life time. Evolution and learning operate on different time scales. Evolution is a form of adaptation capable of capturing relatively slow environmental changes that might encompass several generations, such as perceptual characteristics of food sources for a given bird species. Learning, instead, allows an individual to adapt to environmental changes that are unpredictable at the generational level; think, for example, of the imprinting mechanism that allows some bird species to memorize at birth their own mother [27, 25]. Learning might encompass phenomena like physical development, neural maturation, and synaptic plasticity. Finally, whereas evolution operates on the *genotype* (the genetic specification of an individual), learning directly affects only the *phenotype* (the decoded —embodied— version of the individual), but phenotypic changes cannot directly modify the genotype.

From an engineering point of view, learning is an attractive feature that could be exploited for fast adaptation of an evolutionary machine to unpredictable and contingent circumstances of the environment. Artificial evolution has been recognized as a powerful methodology to develop intelligent robots capable of au-

[1]The authors are with the Microcomputing Laboratory, Swiss Federal Institute of Technology, IN-Ecublens, CH-1015 Lausanne, Switzerland. E-mail: {Name.Surname}@epfl.ch. D.F. acknowledges support by the Swiss National Science Foundation, project nr. 21-49174.26. J.U. is supported by grant nr. BF197.136-AK from the Basque government.

tonomous adaptation to an external environment [8, 18, 35, 38]. An interesting aspect of evolution with respect to other adaptation methods, such as gradient descent techniques or reinforcement learning, is that the criterion function describing the desired behavior needs not be detailed, continuous, and differentiable [16]. On the other hand, artificial evolution can be a very slow process that requires testing of several individuals on the same robot for several generations [29]. In the last few years, some researchers, including ourselves, have attempted to combine local learning methods with artificial evolution in order to achieve faster and more efficient adaptation. In the remainder of this chapter, we shall describe some initial explorations on evolution and learning in simulated and real mobile robots.

1.1.1 Interactions between learning and evolution

Although evolution and learning seem to be complementary and independent mechanisms, it is not yet clear whether these two processes work toward the same goal, whether learning represents an advantage for an evolving system, and how exactly evolution and learning interact. These issues, which have been recently re-examined within a computational framework [5, 48], are relevant for both biologists and engineers.

In the effort to explain evolutionary gaps in the fossil records, more than a century ago James Mark Baldwin [2] advanced the idea that evolution could be influenced by learning during life without assuming that learned features could directly modify the genotype (as hypothesized by Lamarck [11]).[2] Baldwin's argument was that learning accelerates evolution because sub-optimal individuals can reproduce by acquiring during life necessary features for survival. However, since learning requires time (and might thus be a disadvantage), Baldwin suggested that evolution tends to select individuals who have already at birth those useful features which would otherwise be learned. This latter aspect of Baldwin's effect , namely indirect genetic assimilation of learned traits, has been later supported by scientific evidence and defined by Waddington [49] as a *canalization effect* .

The computational advantages of learning in artificial evolution have been made clear only a few years ago by Hinton and Nowlan with a simple model [23]. Imagine a population of chromosomes, each encoding the connection values of a neural network, and a fitness function that is 1 at each life instant only for a few combination of connection values and 0 for all the other combinations. Each gene can have three allelic values: **0** means that the connection is closed, **1** means that the connection is open, and **?** means that the connection can change its value according to a learning mechanism. The learning mechanism employed by Hinton and Nowlan is a simple random process that keeps changing connection weights until —and if— a good combination is found during the limited life time of the in-

[2] Similar views were expressed in the same period by Morgan [34] and Osborn [39].

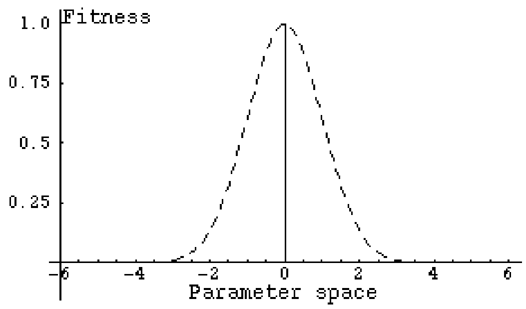

Figure 1.1: Modification of the fitness surface in the Baldwin effect, as suggested by Hinton and Nowlan [23]. In absence of learning, the fitness surface is always zero, except for a few combinations of parameters (thin line in the center). When learning is enabled, there might be several individuals who can reach at some point the combinations that provide fitness points during life, thus enlarging and smoothing out the fitness surface (dotted curve).

dividual. In absence of learning, the probability of finding a good combination of weights would be very small and the fitness surface would look like a flat area with a thin spike in correspondence of the good combinations (figure 1.1). On such a surface a genetic algorithm does not perform better than any random search algorithm. However, if learning is enabled, there are individuals who will eventually achieve one of the good combinations of connection values at some point during life and will thus start accumulating fitness points. This corresponds to an enlargement and smoothing of the fitness area around the good combinations which can be discovered and easily climbed by the genetic algorithm. Individuals with a set of values closer to the good combinations will have higher reproduction probability (because learning reaches faster the condition when it starts accumulating fitness points): the result is that what is initially learned tends to be already present in the genes of the evolved individuals. Despite the ingenuity of the learning mechanism employed, this simple experiment effectively shows how the Baldwin effect might work and its computational advantages.

1.2 Learning and Evolution in Sensorimotor Agents

For the purpose of applying learning mechanisms to an evolving autonomous robot , the experiment performed by Hinton and Nowlan has two major limitations: there is no distinction between the evolutionary and the learning task and the learning mechanism is equated to a random process. In the following two subsections, we shall look at these issues with two experiments on evolutionary artificial organisms equipped with learning abilities. Although both experiments were run in simple grid-world simulations, it was the first time when evolution and learning were considered in the context of artificial organisms equipped with a sensorimotor system.

1.2.1 Learning to predict

If evolutionary and learning goals are the same, one cannot legitimately infer that *whatever* is learned by an individual during life automatically increases the probability of the individual to reproduce. Before jumping to these conclusions, we need more evidence from experiments where evolutionary and learning goals are different.

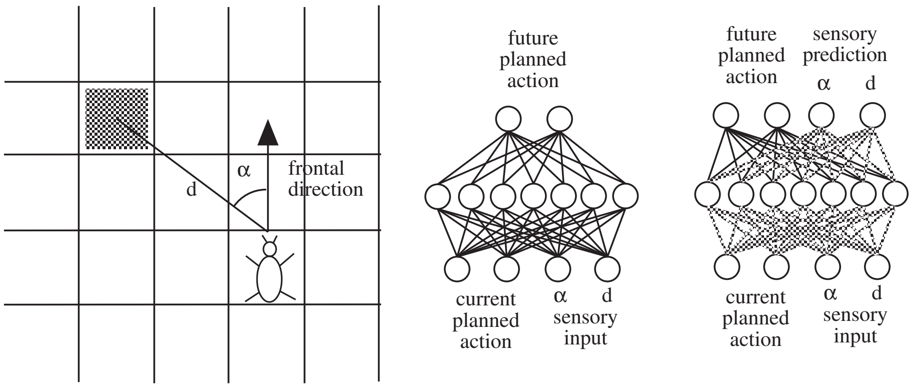

Figure 1.2: **Left**: A portion of the environment with an artificial organism and a food token. The organism perceives the distance and angle of the food with respect to its own frontal direction. **Center**: Neural network architecture evolved without learning. **Right**: Neural network architecture evolved in simulations with learning. Shaded connections are modified by prediction learning during the life of the individual (see text).

Nolfi, Parisi, and Elman [36] have addressed this issue studying populations of simulated organisms interacting with simple environments. Each organism lives in a two-dimensional grid world where a number of food tokens are randomly distributed (figure 1.2). Each food token occupies one cell; if the organism happens to pass on one of these cells, the food token is automatically "eaten" and the organism's fitness is increased. Each individual is equipped with a neural network interfaced to a sensorimotor system that provides input information on the distance and angle (with respect to the facing direction) of the nearest food token and on the planned motor action; the motor system includes four possible actions, coded on two motor units: turn 90° right, turn 90° left, move one cell forward, and stay. At each time step, the neural network receives as input the current sensory information on the nearest food token and the current planned motor action (a^t); the output units produce the next planned action (a^{t+1}). At this point, the planned action that was used as input is executed and the next planned action is passed as new input together with the newly computed sensory information. This sensorimotor cycle is repeated for a number of times (life span) during which the organism is allowed to visit a number of new environments with different food distributions.

The authors have compared two evolutionary conditions, one where the neural network parameters (connection states) are genetically determined and another where each network can also learn during the entire life span of the organism (but without transmitting changes due to learning back into the genotype). In both conditions the evolutionary goal (fitness function) was that of developing organisms capable of eating the highest number of food tokens. The additional learning mechanism employed in the second condition was a gradient descent method (implemented as a back-propagation algorithm [44] of the error between correct responses and neural network outputs), a more efficient solution than random switching of connections used by Hinton and Nowlan. However, the choice of an error correcting mechanism raises the problem of where the correct network output (sometimes called "teaching input") comes from at each time step. It would be technically unfeasible (for a robot) and biologically implausible (for an organism) to assume the presence of an external teacher telling the individual what to do at each time step. A more plausible scenario is learning to predict the sensory consequences of one's own motor actions. Biological and artificial organisms can both decide what actions to take and predict what will their consequences be. An important difference between neural networks applied to a data base (which is by far the most common way of using artificial neural networks in the science and industry) and neural networks of living organisms and autonomous robots is that, in the latter case, the output of the network can affect its own next input [40]. In sensory prediction learning , the teaching signals are readily available from the environment, once the organism has accomplished the action.[3]

In order to incorporate prediction learning in the second condition, Nolfi *et al.* have added to the network two output units (figure 1.2, right side) coding for a prediction of the sensory information after the execution of the current planned action (a^t), and used the freshly-gathered sensory information to update at each time step synaptic weights with the back-propagation algorithm. It should be noticed that although motor actions and sensory predictions are produced by different output units, both systems share the same synaptic weights from the input units to the hidden units.

Three main results were obtained by comparing the conditions with and without learning:

[3] Learning to predict sensory consequences of motor actions has also been recently exploited for autonomous induction of internal models of the interactions between robot and environment [26, 28, 47].

- Learning populations systematically and significantly displayed faster and higher fitness values across generations than populations without learning.
- After a number of generations, individuals learning to predict also increased the ability to find food during life.
- Individuals of the last generation evolved in the condition without learning, when provided with additional output units, could learn to predict better than random individuals (that is, those of the initial generation), indicating that food finding strategies facilitated prediction learning. Furthermore, individuals of the last generations which had evolved in the learning condition, learned to predict even better than those of the no-learning condition, but their prediction performance at birth was identical to random individuals. Therefore, evolution did not select individuals with good prediction abilities, but *individuals with a predisposition to learn.*

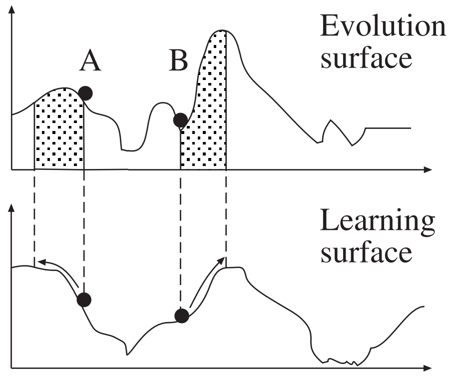

Figure 1.3: Search surfaces when evolution and learning have different goals [36]. y-axes represent evolutionary and learning performances, x-axes represent parameter combinations. Black disks correspond to the initial positions of two different individuals (A and B) on the evolutionary and learning landscapes. Without learning, individual A would have higher reproduction probability. With learning, both individuals tend to move toward higher zones on the learning landscape. The evolutionary fitness of each individual is the integral of the evolutionary area (shaded regions) corresponding to the trajectory in the parameter space caused by learning. These two surfaces are *dynamically* correlated (see text).

Since here the learning criterion is different from the evolutionary goal, the explanation by Hinton and Nowlan depicted in figure 1.1 could not fully explain these results. Nolfi *et al.* have proposed a new explanation. Imagine two different search surfaces , an evolutionary surface and a learning surface (figure 1.3). In the condition where organisms can learn during life, weight change produces a trajectory on the learning surface which corresponds to a displacement on the evolutionary surface. Learning affects evolution in the selection process because the final fitness of the individual will be the sum of the fitness values received during such displacements on the weight space. The correlation between these two search surfaces plays an important role. The authors distinguish between a *static correlation* and a *dynamic correlation*. Static correlation means that maxima and minima occupy the same position in the two surfaces (although they do not necessarily have the same values). Instead, dynamic correlation means that the change in performance (increment or decrement) caused by a certain trajectory is similar in both surfaces (although change rates are not necessarily the same).

From the experimental results summarized above, one can deduce that there is no static correlation between food-gathering and sensory prediction because individuals of the last generations do not have better prediction abilities at birth; however, there is a partial dynamic correlation because after some generations an increment in prediction abilities corresponds to an increment in food-gathering abilities. The authors argue that in populations that both evolve and learn, evolution progressively selects for individuals that are located in regions of dynamic correlation. In other words, evolution selects individuals capable of learning in such a way that they can also have higher reproductive chances.

1.2.2 Evolving how to learn

Another limitation of the experiment by Hinton and Nowlan was the learning mechanism employed. Even assuming that evolution and learning are working toward the same goal (for example, the development of suitable food gathering strategies), biological learning seems to be more directional than random switching of synapses. The experiments conducted by Ackley and Littman [1] in simulated environments not only represent a step forward with respect to this issue, but also introduce a novel self-teaching architecture that could be used right away on real autonomous robots.

A variable-size population of artificial organisms lives in a simulated environment containing predators, different types of food tokens, and other objects. Each organism is controlled by a neural network composed of two modules: an *action module* and an *evaluation module* (figure 1.4), as in classic reinforcement learning architectures [3, 4]. The action module is a feedforward network mapping sensory information into probabilities of executing a certain action; the evaluation module provides an evaluation of the sensory information which is used as a reinforcement signal for the action module. The value of the reinforcement signal depends on the variation of the output of the evaluation module over time. A genetic algorithm evolves the synaptic weight values of both the action and the evaluation modules, but during the life of the individual the action module also modifies the genetically inherited weights using the reinforcement signal provided by the evaluation module. The basic principle behind the weight change rule is that positive reinforcements cause strengthening of active synapses, whereas negative reinforcements cause weakening of active synapses. As in Darwinian evolution , final synaptic weight values are not copied back into the genotype.

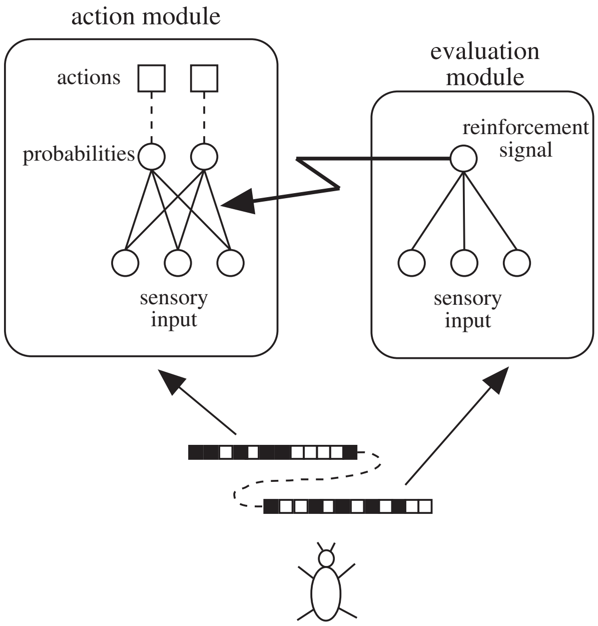

Figure 1.4: *Evolutionary Reinforcement Learning* [1]. The genetic string of an individual encodes the connection strengths of the two modules composing the control system. The action module also modifies its own connection strengths during life, but such changes are not written back into the genetic code.

Each organism can reproduce when it has accumulated enough energy (eating food while escaping predators) and has found another organism ready for reproduction ; offspring are formed by crossing over and randomly mutating the chromosomes of the parents. Organisms die if they cannot maintain a sufficient energy level or when they become too old. As compared to traditional genetic algorithms [24], here the energy level plays the role of an intrinsic fitness function.[4] One way of comparing the effect of different manipulations of the parameters of these artificial world consists in observing how long a population survives. The authors have compared three different situations: evolution without learning , learning without evolution , and the combination of both. The latter condition, also named "Evolutionary Reinforcement Learning" , outperformed both other conditions managing to keep the population alive for over three thousand generations. A longitudinal study of the behavioral characteristics of the artificial organisms showed that during the first 600 generations the action modules were

[4]These types of environments, which are more biologically plausible, are also known as *Latent Energy Environments* [31].

not capable of reaching food at birth, but had to learn it during life. In later generations however, food gathering strategies were already present at birth.

These experiments are similar to the situation described by Hinton and Nowlan because learning and evolution are synergetic mechanisms working toward the same goal. Consequently, here the Baldwin effect manifests itself in its most intuitive aspects: an evolutionary advantage of learning individuals and a gradual genetic assimilation of the features learned during life. The distinctive aspect of the approach employed by Ackley and Littman is that evolution effectively decides how to shape learning, that is when and how to provide positive and negative reinforcements to the action module. In the reinforcement learning literature, the decision of when and how a reinforcement signal of a certain type is provided to the learning system (by the environment) is called a Reinforcement Program (RP). In practice, choosing an appropriate RP makes the difference between a successful learning session and a serious failure. By letting evolution decide under what conditions different reinforcement signals are received by the action module, Ackley and Littman have created an autonomous self-learning system that decides *when* and *what* should be learned. A similar approach has been taken more recently by Nolfi *et al.* [37] in experiments resorting to more realistic simulations of a mobile robot (which will be described below), and showed to be a very effective procedure in dynamic environments that are unpredictable on the evolutionary time scale.

1.3 Evolution of Learning for Autonomous Robots

In the experiments described above, evolution was combined with supervised learning algorithms and all studies were carried out in grid-world simulations. In this section we shall describe a different approach based on the assumption that the neural mechanisms underlying life learning are themselves developed and shaped by the evolutionary process (see section 1.3.1 for biological considerations). A genetic algorithm is used to evolve neural structures that can be continuously modified during life according to the mechanisms specified in the genotype. Each decoded neural network is downloaded into a real mobile robot which is free to interact with the environment while its fitness is automatically computed and stored away for selective reproduction.

We used the miniature mobile robot Khepera which has a circular shape with a diameter of 55 mm, a height of 30 mm, and a weight of 70 g (figure 1.5). Khepera, which has been originally developed at our laboratory [33], is supported by two wheels and two small Teflon balls. Each of two motor controllers sets the speed of its own wheel according to a continuous value between -0.5 and +0.5, where 0.0 means no rotation, -0.5 means maximum rotation speed in one direction (set to 80 mm/s) and 0.5 means maximum rotation speed in the opposite direction. Each of the eight infrared proximity sensors, six positioned on one side and two on the opposite side, returns a continuous value between 0 and 1 that signals the distance

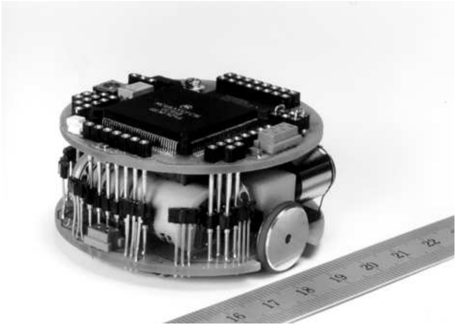

Figure 1.5: The miniature mobile robot Khepera with a ruler. Small black parts around the body are active infrared sensors.

of an object from that sensor (the closer the object, the higher the value returned). In our environment the maximum detection range was approximately 4 cm. The robot was provided with a small positioning device which detected light beams emitted by a laser device placed on the top of the environment and computed the robot absolute position (see [18] for further details). This information was used only for behavior analysis and was not passed to the neural controller. Khepera was attached via a serial port to a Sun SparcStation by means of a lightweight aerial cable and specially-designed rotating contacts.

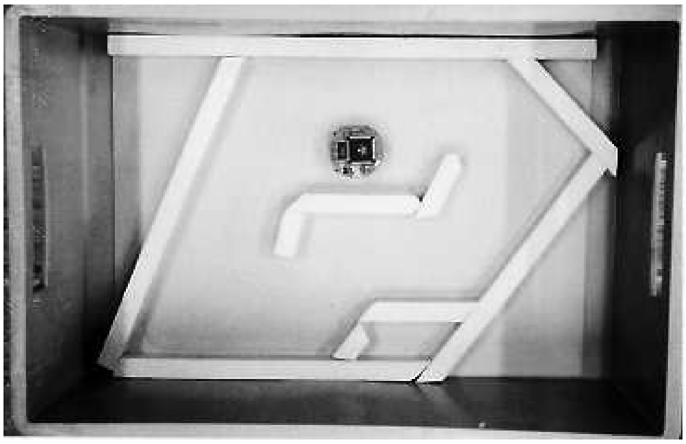

Figure 1.6: Bird-view of the evolutionary environment and the Khepera robot within.

The robot was put in an environment consisting of a circular corridor whose external size was approximately 60x50 cm large (figure 1.6). The genetic operators, the decoding routines from genotypes to phenotypes, and the neural network dynamics were managed by the workstation CPU. Each individual of a population was in turn decoded into a corresponding neural network, the input nodes connected to the robot sensors, the output nodes to the motors (output unit activation was transformed in the range ± 0.5 before passing it to the motor), and the robot was let free to move for 24 s (80 motor actions) while its fitness Φ was automatically recorded at each time step and accumulated. Each sensorimotor loop lasted 300 ms (to/from communications between the robot and the workstation lasted approximately 60 ms) during which the wheel speed was kept constant. Between one individual and the next, a pair of random velocities was applied to the wheels for 5 seconds: this procedure was aimed at limiting the artifactual inheritance of bad locations between adjacent individuals in the population.

The fitness function Φ designed for evolving an obstacle-avoidance and straight navigation behavior

$$\Phi = V \left(1 - \sqrt{\Delta v}\right)(1 - i) \tag{1.1}$$

where $0 \leq V \leq 1$ is a measure of the average rotation speed in absolute value of the two wheels, $\Delta v = (v_{left} + 0.5) - (v_{right} + 0.5), 0 \leq \Delta v \leq 1$ is the absolute value of the difference between the speeds of the wheels transformed into positive values, and $0 \leq i \leq 1$ is the activation value of the proximity sensor with the highest activity. Φ was newly computed every 300 ms, accumulated during the life of the individual, and finally normalized by the number of actions.

A simple genetic algorithm (as described in [20]) was employed to evolve binary chromosomes which encoded a set of parameters describing synaptic properties and learning rules. Every time a new phenotype was created, its synapses were initialized to small random values which could change their strengths, but 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 while the robot interacted with its own environment.

Each neural network had only three neurons, one hidden neuron and two motor neurons, each receiving synaptic connections from all eight infrared sensors and from the hidden neuron (Figure 1.7). Synaptic connections could have a *driving* or a *modulatory* effect on the postsynaptic neuron; afferent signals were combined in a two-component activation function [41] providing an output between 0 and 1. Driving signals determined the direction of rotation of the wheel, whereas modulatory signals could enhance or reduce speed, but could not change the direction of rotation. Each synapse was individually coded on the chromosome by four properties: driving or modulatory (1 bit), excitatory or inhibitory (1 bit), its learning 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 [50]: pure Hebbian, postsynaptic, presynaptic, and covariance (more details are given in [19]).

As soon as the network was decoded and attached to the sensors and motors of the robot, synaptic weight values were initialized to small random values in the

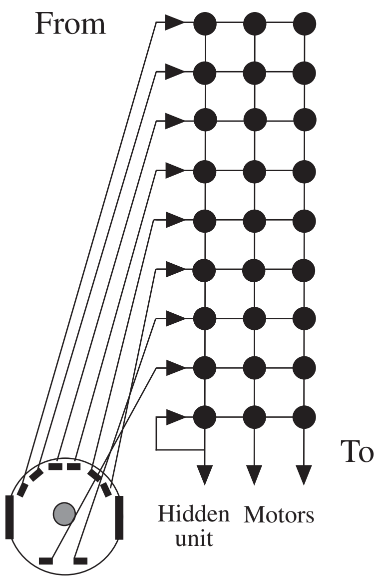

Figure 1.7: The architecture of the neural network employed. Black disks are synapses; the circle in the middle of the robot body represents the hidden neuron.

range $[0.0, 0.1]$ and updated every 300 ms according to the following discrete-time equation

$$w^t = w^{t-1} + \eta \Delta w^t \tag{1.2}$$

where η was the genetically-encoded learning rate, which could assume one of four values $\{0.0, 0.3, 0.7, 1.0\}$.[5]

Three evolutionary runs were made. In all runs the best individual fitness reached a maximum value around the 50th generation ($\Phi = 0.23, \pm 0.09$). All the best individuals of the last generation managed to keep a straight trajectory while avoiding obstacles. The evolved behaviors resulted in smooth paths around the arena (Figure 1.8). This ability was developed by each individual neuro-controller during the first few sensorimotor loops, whatever the initial random values assigned to the synapses. In all the three runs the best individuals of the last generation displayed an emergent frontal direction corresponding to the side where more infrared sensors were placed, which provided a better sampling of the obstacles facing the robot.

Here we provide the analysis of one of the best individuals of the last generation (other individuals had very similar characteristics). Once the neural network had been decoded, connected to the robot sensors and motors, synaptic strengths were initialized to random values in the range $[0.0, 0.1]$, the robot was positioned

[5]If the learning rate is 0.0, the corresponding synapse does not change its strength during the life of the individual.

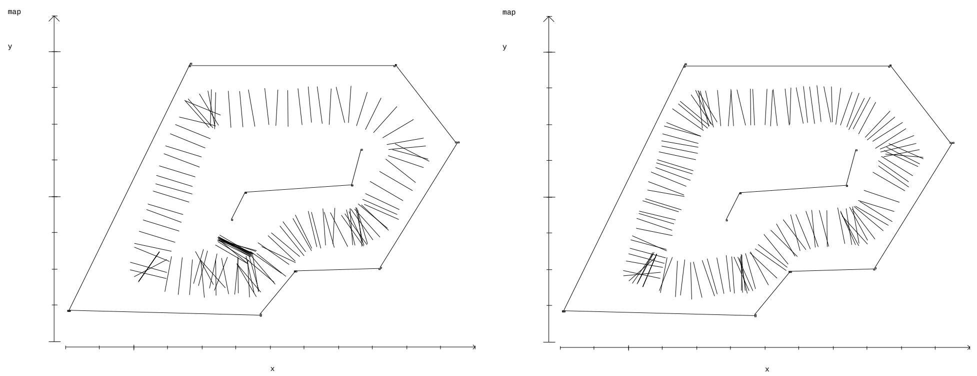

Figure 1.8: Trajectory of an evolved robot learning to navigate during life. Position data, visualized as bars representing the axis connecting the two wheels, were acquired with a laser positioning device every 100 ms. **Left**: trajectory during the first lap (the robot starts in the lower portion of the environment and turns anti-clockwise). **Right**: trajectory at the second lap.

facing a corner of the inner wall (figure 1.8, left; initial position corresponds to the set of superimposed bars in the lower portion of the environment) and let free to move. During the first 2 s (6-7 synaptic updates), the robot adjusted its position alternating back-and-forth movements until it found a wall on its right side. This initial behavior was quite stereotypical: it was displayed for any starting position. Once the wall had been found, the robot moved forward keeping it at a distance of 2 cm from its own right side; every second or third action, it slightly turned toward the wall and then continued forward. This sort of jerky behavior was gradually reduced when coasting a straight long wall (e.g., the north and east walls). If the wall was slightly bent, the robot could still follow it without reducing speed, but when it encountered a convex angle smaller than 90° (which means that most of the front infrared sensors are active) the robot stopped, backed rotating to the right, and then moved forward in the new direction. After one lap around the maze, the path became smoother with less trajectory adjustments and more tuned to the geometric outline of the environment (figure 1.8, right).

1.3.1 Fast changing synapses

The development of such behavior can be understood by studying the internal dynamics of the evolved network. Figure 1.9 plots the strengths of all synapses in the network during the first 100 actions (sensory-motor loops) visualized in Figure 1.8 using the same format of Figure 1.7. Without going into much detail (which can be found in [19]), one of the most remarkable results is that synapses continuously change while the behavior is quite stable. In the conventional view, synapses are relatively slow-changing and stable components of

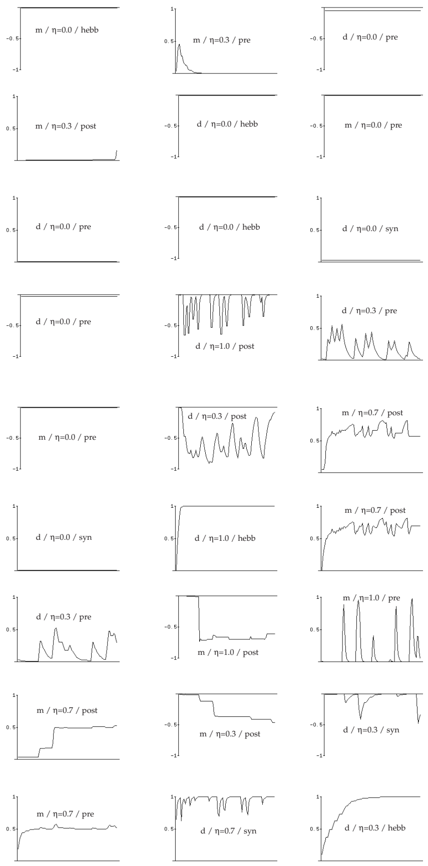

Figure 1.9: Synaptic strength recorded every 300 ms during the first 100 actions of the robot. y-axis position indicate excitatory (pointing upwards) or inhibitory synapses (pointing downwards).

the nervous system. Synaptic change is identified with learning of new skills or acquisition of new knowledge, while neuron activation is identified with behavior expression. Typically, acquisition of a stable behavior in a static environment corresponds to stability (no further change) of individual synapses (e.g., see [22]). 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 is not included into the fitness function employed in this experiment, which is defined only in behavioral terms.

The functioning of our system offers a complementary —but not necessarily alternative— explanation. Synapses are responsible for both learning and behavior regulation. Knowledge in the network is not expressed by a final stable state of the synaptic configuration, but rather by a dynamic equilibrium point in

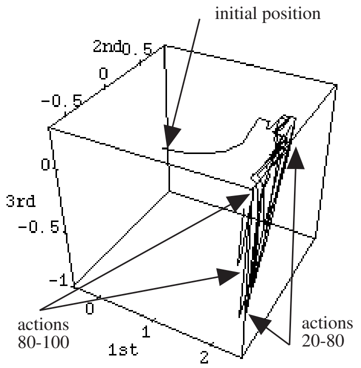

Figure 1.10: State-space representation of synaptic dynamics during the first 100 actions plotted as trajectory within the space of the first three principal components. Arrows indicate starting position and range of oscillation between action sequences 20-80 and 80-100. Oscillations within the subspace of the third (smallest) component correspond to fine trajectory adjustments. Method: Sanger's network [45] for extracting the first three principal components of the input correlation matrix was trained to stability on the 27-component vectors corresponding to the synaptic activity recorded during the first 100 actions of the robot (visualized in Figure 1.9); after training, input vectors were presented again to the network and output unit activations were plotted in the resulting 3-dimensional space.

an n-dimensional state-space (where n is the number of synapses). Figure 1.10 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 1.8. During the first six actions 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 slice of space corresponding to the third principal component corresponds to fine trajectory adjustments and is further reduced as the robot gradually tunes its path to the geometry of the environment. This means that, after an initial phase of adjustment, the synapses as a whole change in a systematic and co-ordinated fashion. In other words, the stable behavior acquired during life is regulated by continuously changing synapses which are *dynamically stable*.

From an engineering point of view, evolution of learning abilities , that is of robots capable of autonomously self-reconfiguring their internal parameters on the fly depending on external conditions, represents a good trade-off between the long time required for artificial evolution and the flexibility of the evolved system. However, one of the limitations of this experiment is that it has been tested only in a static environment where evolution alone could come up with similarly efficient solutions [17].

1.4 Adaptive Behavior in Co-evolutionary Competitive Agents

Adaptive behavior during life –as compared to innate and fixed behavior– might represent an advantage in unpredictable and dynamic environments. In this respect, co-evolution of competitive species provides an interesting testbed to study the role of adaptive behavior. In the simplest scenario of two competing species, such as a predator and a prey, the behavior of each individual is tightly related to the behavior of the competitor both on the evolutionary and on the ontogenetic time scale. On the evolutionary time scale, the coupled dynamics of the system give rise to the "Red Queen effect" whereby the fitness landscape of each population is continuously modified by the competing population [10].

On the ontogenetic time-scale, it has been argued that pursuit-evasion contests might favour the emergence of "protean behaviors", that is behaviors which are adaptively unpredictable [14]. For example, prey could take advantage of unpredictable escape behaviors based on short sequences of stochastic motor actions. Similarly, predators could take advantage of enhanced perceptual characteristics and/or adaptive sensory-motor intelligence which could enable predictive tracking strategies (for a good introduction, see [32]). However, none of these experimental studies systematically explored the role of ontogenetic adaptive behavior in co-evolution of competing species. Although most of the evolved systems include some form of noise, it is difficult to say whether this plays an important role on the specific dynamics of co-evolving species or it is simply exploited for smoothing the fitness landscape. All the results presented so far in the literature are based on single-run studies and do not include comparisons between different adaptation techniques.

We decided to look at life learning in co-evolutionary systems using an approach similar to that described in the previous section. In particular, we were interested in understanding whether life-learning might change co-evolutionary dynamics and what type of adaptive behavior each species (predator and prey) would exploit. 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 1.11). The two species evolved 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.

Since evolutionary experiments take a long time on real robots, before implementing our system on the Khepera robots, we decided to systematically study the dynamics in realistic simulations (figure 1.12).[6] The controllers of the two

[6]The evolved controllers were then downloaded on the physical robots. Furthermore, the

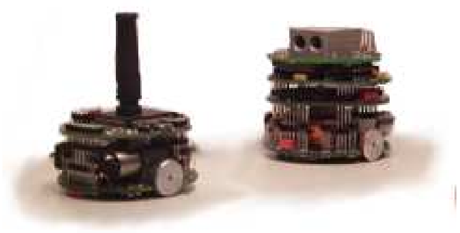

Figure 1.11: **Right**: The Predator is equipped with the vision module (1D-array of photo-receptors, visual angle of 36°). **Left**: The Prey has a black protuberance which can be detected by the predator everywhere in the environment, but its maximum speed is twice that of the predator. Both Predator and Prey are equipped with 8 infrared proximity sensors (max detection range was 3 cm in our environment).

robots were simple perceptrons with two sigmoid units and recurrent connections at the output layer. The activation of each output unit was used to update the speed value of the corresponding wheel every $100ms$. In the case of the predator, each output unit received connections from five visual units, each corresponding to the activity of 7 physical photoreceptors, and from eight infrared proximity sensors; in the case of the prey, each output unit received input only from 8 infrared proximity sensors, but its activation value was multiplied by 2 before setting the wheel speed.

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 [46, 43, 9]) in order to improve co-evolutionary stability. A tournament ended either when the predator touched the prey or after 500 motor updates (corresponding to 50 seconds on the real robots). The fitness function Φ_c for each tournament c was simply *TimetoContact* normalized by the maximum number of motor updates (500) TtC for the predator pr, and $1 - TtC$ for the prey py, further averaged over the number of tournaments (10). Therefore the fitness values were always between 0 and 1, where 0 is worst.

Three different co-evolutionary conditions have been studied, each one corresponding to a different controller type (Table 1.1), but all using the same architecture and chromosome length. In all conditions, the first bit of each synapse coded its sign (whether excitatory or inhibitory). In the first condition, the remaining four bits coded the synaptic strength as a value in the range $[0, 1]$: since no

most interesting evolutionary runs were replicated directly on the physical robots.

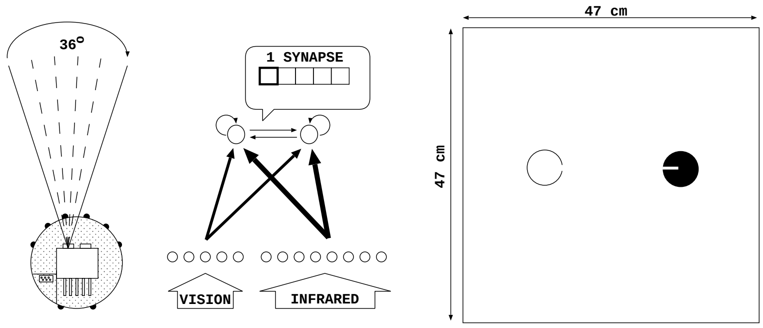

Figure 1.12: **Left and center**: Details of simulation of vision, of neural network architecture, and of genetic encoding. The Prey differs from the Predator in that it does not have 5 input units for vision. Each synapse in the network is coded by five bits, the first bit determining the sign of the synapse and the remaining four the other synaptic parameters. **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.

changes take place during the life of the individuals, let us call them *genetically-determined controllers* . In the second condition, only two bits coded the synaptic strength (again, in the range $[0, 1]$), and the remaining two bits coded the level of random noise applied to the synaptic value. Each level corresponded to the lower and upper bounds of a uniform noise distribution: 0.0 (no noise), ± 0.337, ± 0.667, and ± 1.0. At every network activation, each synapse had its own newly-computed noise value added to its strength (with a final check to level out sums below 0.0 or above 1.0). We shall call this condition *adaptive-noise controllers* because each species can evolve the most appropriate noise level for each synapse. In the third condition, two bits coded four Hebbian rules and the remaining two bits the learning rate (0.0, 0.337, 0.667, and 1.0), exactly as described above

	Bits for one synapse				
Condition	1	2	3	4	5
1	sign	strength			
2	sign	strength		noise	
3	sign	Hebb rule		rate	

Table 1.1: Genetic encoding of synaptic parameters for each co-evolutionary condition. 1: Genetically-determined controllers; 2: Adaptive-noise controllers; 3: Directional-change controllers.

in section 1.3. Let us call this condition *directional-change controllers* , simply indicating that synaptic changes depend on sensory activation and motor actions.

For each condition, 6 different evolutionary runs were performed, each starting with a different seed for initializing the computer random functions.[7]

Protean behaviors

As for biological competitive systems [42], population performances display oscillatory behaviors in all three conditions (figure 1.13). However, here nominal fitness is not an indicator of fitness progress. For example, a reduced fitness of the predators does not necessarily mean that their behaviors have deteriorated; in fact, it might be explained by behavioral progress achieved by the prey. A

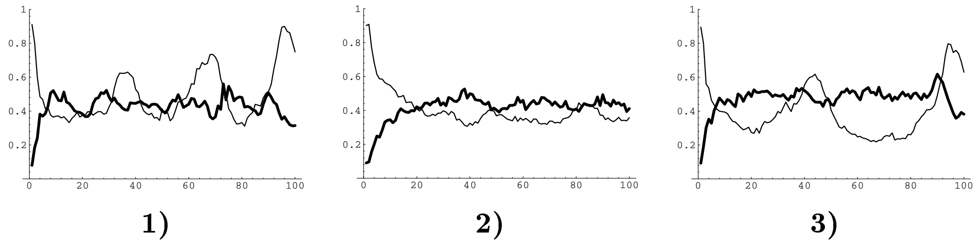

Figure 1.13: Average population fitness across generations for the three different conditions. Thick line is predator, thin line is prey. 1) Genetically determined; 2) adaptive noise; 3) directional change.

relational measure of performance gives us additional information on the coupled dynamics of such 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 competing species at each generation for each separate run i in a specific condition c. In our co-evolutionary runs which lasted 100 generations, such index will be in the range $[-100, 100]$, where -100 means that the prey 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 prey.

For the genetically-determined condition ($c = 1$), 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. Similar results were measured in the condition with evolutionary adaptive noise ($c = 2$) which displayed an average relative performance $\overline{R^2} = 11.66$, with standard deviation of

[7] 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. Therefore, for each condition below, we shall plot only data referring to seed 1 (arbitrarily chosen), but the statistical tests reported will be based on all runs.

the sample mean $\sigma = 32.5$ (probability value was 0.83 for t-test of the difference of the means between the two conditions). However, relative performance of the two species in the directional-change condition significantly differed from both other conditions, $\overline{R^3} = 72$ 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 this last condition, the predators profited of the learning ability and reported almost always better performance than the prey. Furthermore, in all runs, the average fitness of the predator population was more stable than that of the prey.

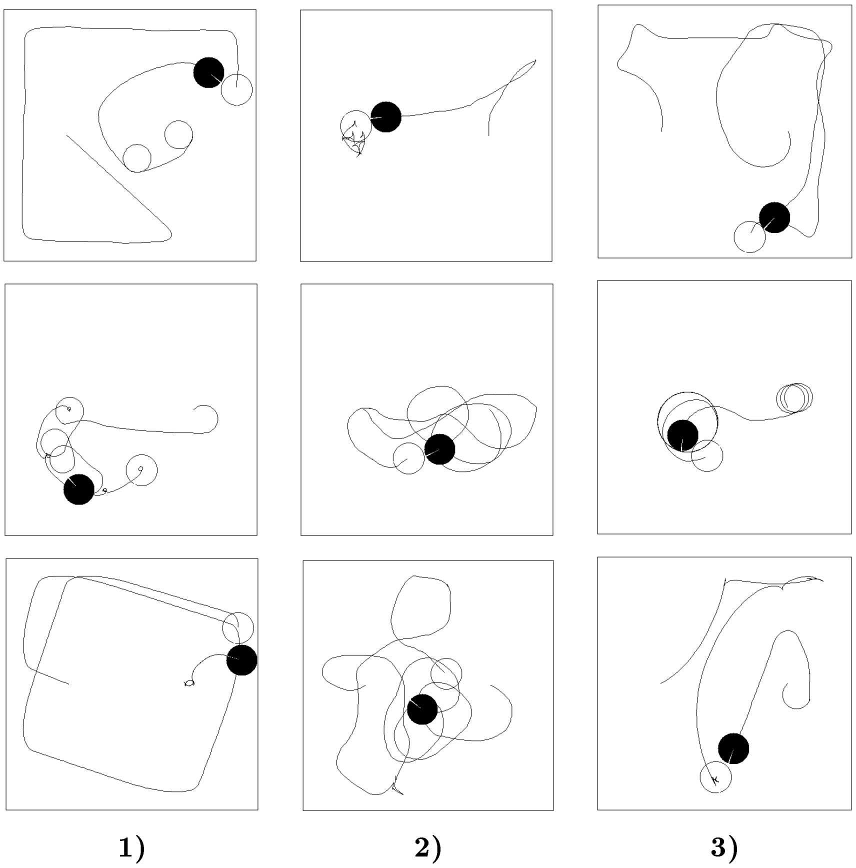

Figure 1.14: Interesting behaviors recorded at different points in evolutionary history (from top to bottom) for each condition. Black disk is predator, white is prey. 1) genetically determined (generations 20, 70 , 90); 2) adaptive noise (generations 20, 50 , 80); 3) directional change (generations 20, 70 , 95).

A more intuitive understanding of the results can be gained by looking at typical behaviors of the two species recorded at interesting points of co-evolution (figure 1.14). When the individuals cannot modify synaptic weights during life, several different behavioral strategies rapidly evolve and alternate over generations for both individuals, but none of these strategies can always win over those developed by the competitor. For example, in the first row of figure 1.14.1, we have a prey that follow walls and a predator that has developed good visual tracking abilities. Later on, as shown in the second row, the prey waits for the predator and rapidly avoids it when it gets too close. After a few more generations the prey resumes a sort of wall-following behavior, but the predator develops a "spider strategy": it slowly backs against a wall and there it waits for the fast approaching prey which detects its enemy too late to avoid it.

The introduction of modifiable controllers in co-evolutionary competition significantly affected various aspects of the system dynamics, both on the evolutionary and on the ontogenetic time-scale. On the evolutionary time-scale, noisy controllers ($c = 2$) caused a relaxation of the tightly coupled dynamics observed in the benchmark condition ($c = 1$). High behavioral variation during life of the competitors, especially in the case of prey, was such that only a sufficiently general behavioral strategy could pay off, both for the predator and for the prey. Co-evolutionary search here had higher probability of selecting individuals located in better zones of the fitness landscape, a mechanism very similar to that explained by Hinton and Nowlan. Instantaneous fitness values reflected more closely behavioral progress, as in traditional single-agent static environments. On the other hand, directional-change controllers ($c = 3$) clearly favored dominance of one species —the predator— whose sensorimotor system profited most of non-random changes of synaptic values.

On the ontogenetic time scale, that is at the level of individual tournaments, the two species differentially exploited the two types of modifiable controllers. In condition 2, both species reported similar performances, but they differently exploited adaptive noise: prey employed higher noise levels to generate unpredictable and hard-to-track trajectories, whereas predators reduced noise level to maintain sufficient pursuit strategies (figure 1.14.2). In condition 3, predators benefited from directional synaptic change to improve their pursuit abilities with respect to condition 1. As one can see in figure 1.14.3, there is not much variation in the behavior of the predator. It always displays a very good tracking ability across generations: once the prey has been locked in its visual field, it quickly accelerates to maximum speed until contact. In this condition, for the predator it is sufficient to get the sign of the synapses right. Then, independently of their initial random values, the synapses from active sensors will be increased causing an acceleration in the right direction. 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 would have required accurate settings of all synaptic strengths from visual inputs. Here instead, since synapses are temporarily increased depending on active units, adjustments take place when and where required depending on current sensory input. The trajectory in the second row of figure 1.14.3 displays an example of synaptic adjustment. 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).

1.5 Local and Global Adaptive Search in Modular Architectures

If from a biological and epistemological point of view it is worthwile exploring with real robots the implications of our notion of phylogenetic and ontogenetic adaptation, from an engineering point of view it is necessary to take a set of short-cuts and devise novel strategies for the specification and development of adaptive robots that comply with standard engineering methods and practical constraints.

Modern approaches to behavior engineering of autonomous robots have stressed the importance of modular and distributed architectures composed of simple and interconnected elements [6, 12] where each component has full or partial access to sensory data and can affect the actions taken by the robot. Distributed modular control has several potential advantages: it is an open system, it is intrinsically robust to local failures, and it is suitable for gradual "shaping", i.e. incremental training of independent behavioral competences [13].

In several practical situations, it is neither necessary nor advisable to train a control system completely from scratch. On the one hand, learning not only takes long time, but it also does not guarantee to converge if the parameter search space is very large. On the other hand, there might be some aspects of the behavior that could be easily pre-programmed exploiting available knowledge about the task constraints. A modular architecture can easily accommodate both adaptive and pre-programmed modules (the latter are sometimes referred to as "instincts").

Adaptation in modular architectures can take place at two different levels: within each individual module and between modules. The first level is often concerned with acquisition of a new competence for a partially defined module and/or with re-adaptation of some of its parameters, such as threshold tuning, to accommodate minor changes in the environment or robotic platform. This aspect of adaptation is equivalent to a local search in a restricted parameter search and is well-suited for learning mechanisms such as –for example– reinforcement learning. The second level of adaptation is concerned with coordination of all modules and/or with acquisition and integration of new modules to cope with major environmental changes and task constraints. This aspect of adaptation is analogous to a global search over a less granulated space composed by a finite set of behavioral competences, and seems well-suited for an incremental and evolutionary approach. A similar proposition has been advanced in [7], where the possibility of evolving high-level behavior primitives was explored.

We have experimented with an open modular architecture which can be incrementally shaped via evolutionary and learning mechanisms while the robot interacts with its own environment. In doing so, we had three main concerns:

- the architecture must be capable of integrating adaptive modules with pre-programmed behaviors, but also be able to discard them latter if they become inefficient;
- the system must allow for incremental and autonomous construction of a suitable architecture, as demanded by the shaping policy or by major changes in the task constraints;
- individual modules must be capable of quickly re-adapting themselves to local changes without requiring a full redesign of the whole architecture.

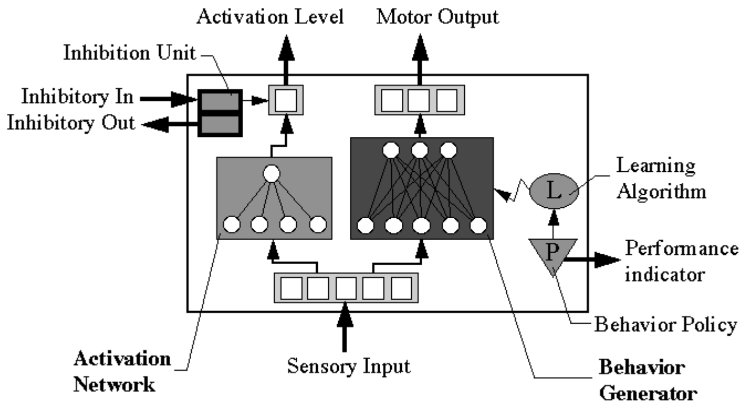

Figure 1.15: Structure of a behavioral module. The behavior generator shown here is a neural network, but it could be any other structure, such as a pre-designed and fixed program. The output of the behavior generator can be a motor command or any other decision. Sensory inputs may include decision signals from other modules. In the current implementation, links between modules can be only inhibitory or absent.

Application of artificial evolution requires a proper definition of the elements on which it operates, in this case individual modules, so that the decoded genotype corresponds to a legal and meaningful architecture. Each module (figure 1.15)of the architecture receives input from the sensors of the robot and from other modules, and sends output signals to other modules and to the motors of the robot. The internal structure is based on two components: an *activation network* and a *behavior generator*. The activation network decides on the basis of current (and/or previous) sensory information whether the module becomes active. The behavior generator can be a pre-programmed behavior, a neural network (as in figure 1.15), a classifier system, or any other structure capable of generating motor commands or other behavioral decisions in response to sensory inputs. If the behavior generator is capable of adaptation, the module includes a suitable local learning algorithm and a learning policy. In the experiment described below, the behavior generator of some modules consisted in a neural network adapting through a self-supervised reinforcement learning algorithm.

A module can send and receive signals from other modules in two ways: a signal can be a piece of information, such as a decision output or input, or a link to decide which module(s) should be activated at every time step. The former type of signal is basically exchanged between the behavior generators, whereas the latter is exchanged between activation networks. In the current implementation, activation links between modules can be only inhibitory, where they exist. If a module becomes active, it attempts to inhibit all modules to which it is connected; similarly, it receives inhibitory signals from other connected active modules. The module winning this competition will access the motor resources and control the robot for a certain time slice.[8] The pattern of connectivity among the modules and their individual activation networks are encoded as a binary string and evolved by a genetic algorithm.

Evolution is incremental and operates on variable-length genotypes . Initially, a set of basic modules are defined on the basis of general knowledge about the task requirements and robot sensors. The activation networks of the modules and the connectivity pattern among modules define the genotype length. An initial population of such controllers is evolved until an individual is generated that satisfies the task criteria. Individual modules can be individually trained before evolution and/or during evolution, depending on the task constraints. If the task constraints change, or if new hardware modules are added to the robot, it is possible to define new modules and increment the genotype length by including the new activation networks and all the connectivity to previous modules. However, old parts of the genotype can be masked so that they cannot be crossed over or mutated. Incremental evolution is useful not only for changing environments and for variable robot configurations, but also for progressively achieving very complex behaviors. It has been shown that by gradually increasing environment complexity [15] and modifying the fitness function [21], it is possible to evolve behaviors which otherwise would not be evolvable.

1.5.1 Battery recharge and object collection

The methodology described above was tested for the development of a complex task in a changing environment. A Khepera robot equipped with an additional gripper module, was positioned in an environment including several small objects, a battery charger and a light source positioned above it (figure 1.16). Instead of

[8] It is easy to extend the current version to the case of excitatory and inhibitory links with variable strengths, which makes the global architecture similar to a recurrent neural network.

using the real batteries available on the Khepera (which last approximately 30 minutes and require additional 40 minutes to recharge), a virtual battery lasting between 40 and 60 seconds (depending on motor activity) and a fast virtual recharger (taking 5 seconds to recharge) were employed during training. This solution did not change the difficulty and/or realism of the training environment, but considerably speeded up our measures.

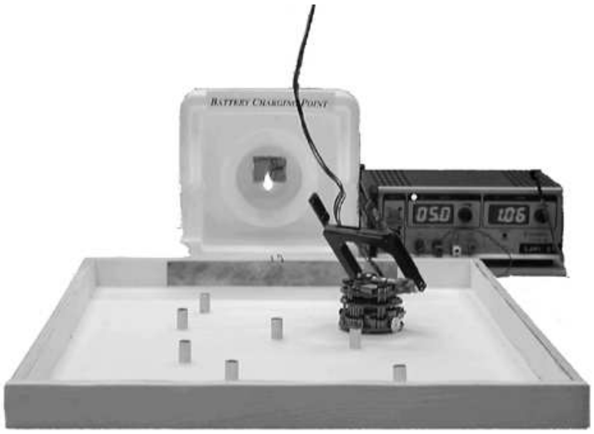

Figure 1.16: The environment used for the object collecting experiment. The battery charger and the light source are visible on the background.

In order to test several parts of our methodology, we devised two different tasks to be achieved sequentially and incrementally. As a first goal, we attempted to evolve a controller capable of moving the robot around the arena without hitting obstacles and returning to the recharging station before full discharge of the batteries. At this stage, the gripper module was not yet plugged on the top of the robot. Four initial modules were defined (see figure 1.15):

Wander Used to move to robot around the environment. The behavior generator has been implemented both as a pre-designed program and as an adaptive neural network because different wandering behaviors might be used depending on the tasks requirements. For our experiment, it turned out that simple straight motion was learned in few seconds and was perfectly adequate for the task. Input: proximity sensors. Output: wheel commands.

Follow-Light It goes toward the light source moving the robot in the direction of the population vector resulting from the activity of all ambient light sensors. No learning mechanism is required here because there is only one light source and sufficient gradient information in the environment. Input: ambient-light sensors. Output: wheel commands.

Recharge When active, it freezes every motor activity until the battery charge indicator is on the full zone. Input: front proximity sensors and battery charge indicator. Output: all available motors.

Obstacle-Avoidance The behavior generator is an adaptive neural network mapping sensor activations into one of four possible motor actions (go forward, turn right, turn left, move backward). Input: proximity sensors. Output: wheel commands.

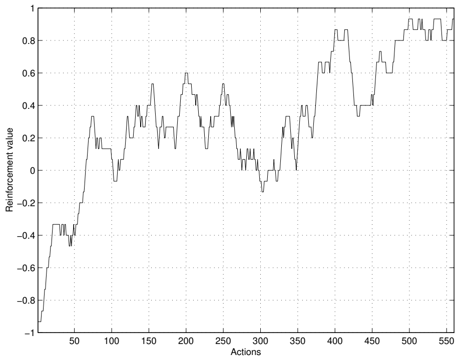

Figure 1.17: Performance of the **Obstacle-Avoidance** behavior generator during reinforcement learning before evolution. One action lasts approximately 100 ms. Reinforcement values $-1, 1$ are averaged over a moving window (window size = 30). Performance below zero means higher percentage of negative reinforcements, above zero higher number of positive reinforcements.

Both the **Wander** and **Obstacle-Avoidance** modules were separately trained before evolution, but the learning process could be locally and automatically resumed at any time during evolution if the performance indicators reached a critical level (in case that some characteristics of the environment or of the mechanical components changed). For each adaptive module, a local reinforcement learning algorithm [30] attempted to satisfy a simple behavior policy exploiting only information locally available to the module. For example, for the **Obstacle-Avoidance** module, the policy was the minimization of the activation level of the proximity sensors. Since each module is relatively simple and operates only on local information, the learning process is very fast, as shown in figure 1.17.

Once existing adaptive modules have been trained, a population of chromosomes, each encoding the activation networks and the connectivity pattern between modules, are randomly initialized and evolved on the Khepera using the

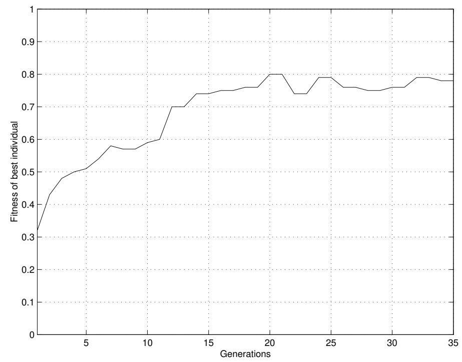

Figure 1.18: Evolutionary training of the controller for exploration and battery charging task (maximum fitness which can be obtained without recharging is approximately 0.55). Fitness 1.0 could be theoretically achieved only in an environment without objects and walls.

procedure and fitness function already described in section 1.3. Fitness values are accumulated at each time step; therefore, robots that are capable of periodically recharging will totalize higher fitness scores. A maximum limit of 200 actions, roughly corresponding to twice the battery life span, was allowed for all individuals. After only 20 generations, there are individuals who are capable of performing the desired task (figure 1.18).

In a second stage, we plugged the gripper module on the Khepera and we changed the task definition. Now the desired behavior was that of collecting the highest number of objects and releasing them outside the arena, recharging the batteries when necessary. The gripper module has two degrees of freedom (figure 1.16): it can lift/lower the arm and open/close the gripper. An optical barrier between the two segments of the gripper provides sensory information on the presence of an object. Three additional modules were defined as follows:

Object-Identification Used to discriminate between graspable objects and obstacles. The behavior generator is an adaptive network which autonomously learns by exploring a detected object. Input: proximity sensors, optical barrier. Output: wheel commands, gripper commands, object class.

Object-Grasp If the Object-Identification module recognizes a graspable object, and if the optical barrier is off, it moves toward the direction of the population vector resulting from the activation of the proximity sensors, it lowers the gripper, it backs until the optical barrier is on (object well-positioned), closes the gripper, and lifts the object. The module is not adaptive. Input: optical barrier, Object-Identification output, proximity sensors. Output: wheel commands, gripper commands.

Object-Release If the Object-Identification module recognizes a wall, it moves toward the population vector resulting from the activation of the proximity sensors, lowers the gripper, and drops the object. Input: optical barrier, Object-Identification output, proximity sensors. Output: wheel commands, gripper commands.

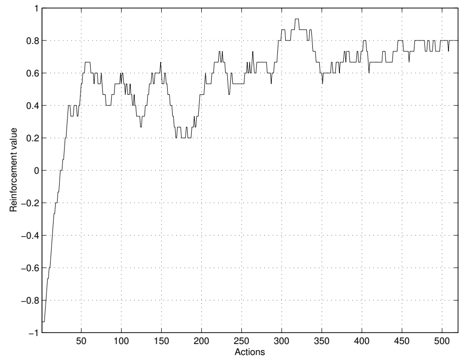

Figure 1.19: Performance of the **Object-Identification** module during reinforcement learning on small objects (diameter is 10 mm). One action lasts approximately 3 s. Reinforcement values $-1, 1$ are averaged over a moving window (window size = 30). Performance below zero means higher percentage of negative reinforcements, above zero higher number of positive reinforcements. Performance around zero means random discrimination.

The **Object-Identification** module learns to discriminate objects by exploring them with the gripper. If some sensors are active, it formulates a hypothesis on the object type; then, it moves the robot toward the direction of the population vector resulting from the sensor activity and it lowers the gripper while backing a little bit. If the encountered object is graspable, then the light barrier becomes on. The signal coming from the optical barrier (off or on) is used to provide the reinforcement value and learn the sensory configuration associated to the corresponding object. As for other adaptive modules, learning employs only information locally available to the module, is carried out separately, but it can be automatically resumed during evolution if the performance indicator reaches a critical threshold. After approximately 20 minutes of training with small metallic objects (all having the same diameter of 10 mm) scattered around the arena, the module is capable of reaching a satisfactory stable performance.

In order to evolve the full architecture, the genotype size is increased by allocating new bits for all the new activation networks and links among new and old modules. The old parts of the genotype are masked so that they are not affected by crossover and mutation. The fitness function is augmented with an additional component that gives 0.5 points for every grasped object and further

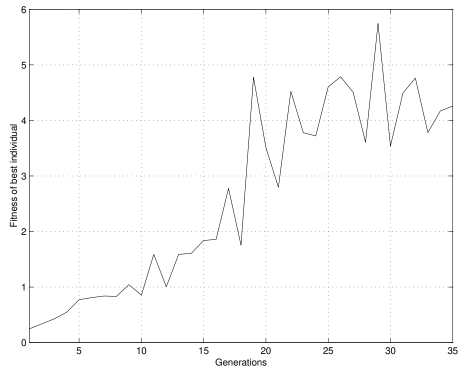

Figure 1.20: Evolutionary training of the controller on the full task, including object picking and releasing. Fitness values above 1.5 indicate that at least one object has been picked up and correctly released outside the arena. Given the limit put on the life duration of each individual, a robot can collect and release 5 objects at best, including a battery recharge, during the life span.

0.5 points if the object is correctly released. After a few generation of incremental evolution (figure 1.20), the best individuals are capable of executing the complete final task: collecting objects and recharging when necessary.

As stated above, one of the advantages of a modular architecture is that individual modules can autonomously and locally re-adapt to small environmental changes without requiring re-training of the complete architecture. After evolutionary training, we have substituted all objects with larger ones. The new objects, which have a diameter (25 mm) almost twice that of the previous ones, are often recognized as walls (see initial performance drop in figure 1.21 as compared to final performance reported in figure 1.19). However, the **Object-Identification** module automatically and quickly adapts the synaptic weights to the new object size without external supervision (figure 1.21).

1.6 Conclusion

In this chapter we have examined the relationship between evolution and learning from the computational perspective of autonomous sensorimotor agents. The general message emerging from the models and experiments described above is that the combination of these two adaptive methods is more than a sum. Evolution and learning interact affecting each other's search space and search modality. Learning can ameliorate and accelerate evolution by modifying the selective-reproduction probability of evolving individuals. Evolution decides what are the most suitable learning mechanisms and how learning operates. Finally, learning is used to adapt to local and relatively fast changes in the environment which are

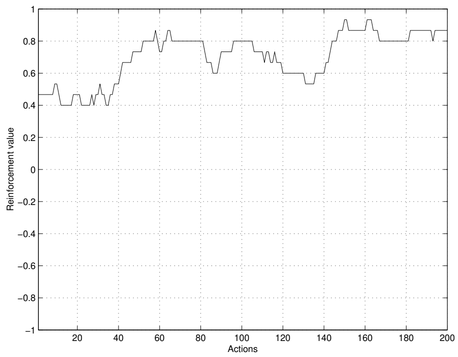

Figure 1.21: Performance of the **Object-Identification** module during reinforcement learning when small objects are replaced with new larger objects (diameter is 25 mm). Initial performance is given by the generalization from the previously learned synaptic weights. One action lasts approximately 3 s. Reinforcement values $-1, 1$ are averaged over a moving window (window size = 30). Performance below zero means higher percentage of negative reinforcements, above zero higher number of positive reinforcements. Performance around zero means random discrimination.

unpredictable on the evolutionary scale.

Computational explorations of interactions between evolution and learning, conducted in laboratories of artificial life where several variables can be manipulated as desired and their effects systematically studied, provide important insights for a better understanding of the principles of biological adaptation. On the other hand, the combination of evolution and learning can be fruitfully exploited for engineering robots equipped with improved behavioral autonomy and self-adaptation mechanisms.

It is difficult to assess to what extent our computational notion of learning and evolution reflects biological mechanisms. In any case, the biological inspiration behind the adaptive algorithms employed, such as genetic algorithms and artificial neural networks, represents a rich source of new ideas that can be rigorously tested and exploited for the realization of novel and robust machines.

Bibliography

[1] D. H. Ackley and M. L. Littman. Interactions between learning and evolution. In C.G. Langton, J.D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life II: Proceedings Volume of Santa Fe Conference*, volume XI. Addison Wesley: series of the Santa Fe Institute Studies in the Sciences of Complexities, Redwood City, CA, 1992.

[2] J. M. Baldwin. A new factor in evolution. *American Naturalist*, 30:441–451, 1896.

[3] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):835–846, 1983.

[4] A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Learning and sequential decision making. In M. Gabriel and J. W. Moore, editors, *Learning and Computational Neuroscience*, pages 539–602. MIT Press-Bradford Books, Cambridge, MA, 1990.

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

[6] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[7] R. A. Brooks. Artificial Life and real robots. In F. J. Varela and P. Bourgine, editors, *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life.* The MIT Press/Bradford Books, Cambridge, MA, 1992.

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

[9] 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.

[10] R. Dawkins. *The Blind Watchmaker.* Longman, Essex, 1986.

[11] J. B. P. A. de M. de Lamarck. *Zoological Philosophy.* MacMillan, London, 1914. Relevant excerpt reprinted in [5].

[12] M. Dorigo and M. Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71:321–370, 1994.

[13] M. Dorigo and M. Colombetti. *Robot shaping: An experiment in behavior engineering.* MIT Press, Cambridge, MA, 1998.

[14] P. Driver and N. Humphries. *Protean behavior: The biology of unpredictability.* Oxford University Press, Oxford, 1988.

[15] D. Floreano. Emergence of Home-Based Foraging Strategies in Ecosystems of Neural Networks. In J. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *From Animals to Animats II: Proceedings of the Second International Conference on Simulation of Adaptive Behavior.* MIT Press-Bradford Books, Cambridge, MA, 1993.

[16] D. Floreano. Reducing human design and increasing adaptivity in evolutionary robotics. In T. Gomi, editor, *Evolutionary Robotics*, pages 187–220. AAI Books, Ontario, Canada, 1997.

[17] 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*, pages 402–410. MIT Press-Bradford Books, Cambridge, MA, 1994.

[18] 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.

[19] D. Floreano and F. Mondada. Evolution of plastic neurocontrollers for situated agents. In P. Maes, M. Matarić, 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*, pages 402–410. MIT Press-Bradford Books, Cambridge, MA, 1996.

[20] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, Redwood City, CA, 1989.

[21] 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.

[22] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley, Redwood City, CA, 1991.

[23] G. E. Hinton and S. J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[24] J. H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, 1975.

[25] K. Immelmann. The evolutionary significance of early experience. In G. Bearends, C. Beer, and A. Manning, editors, *Function and Evolution in Behaviour*, pages 243–253. Clarendon Press, Oxford, 1976.

[26] M. I. Jordan and D. E. Rumelhart. Forward Models: Supervised Learning with a Distal Teacher. *Cognitive Science*, 16:307–354, 1992.

[27] K. Z. Lorenz. The Evolution of Behaviour. *Scientific American*, 199(6):67–78, 1955.

[28] M. Marjanović, B. Scassellati, and M. Williamson. Self-taught visually-guided pointing for a humanoid robot. In P. Maes, M. Matarić, 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*, pages 36–44. MIT Press-Bradford Books, Cambridge, MA, 1996.

[29] M. Matarić and D. Cliff. Challenges in Evolving Controllers for Physical Robots. *Robotics and Autonomous Systems*, 19(1):67–83, 1996.

[30] L. Meeden. Developing Neural Network Controllers for Robots. *IEEE Transactions on Systems, Man and Cybernetics: Part B: Cybernetics*, 26:474–484, 1996.

[31] F. Menczer and R. K. Belew. Latent energy environments. In R. K. Belew and S. Mitchell, editors, *Plastic Individuals in Evolving Populations*. Addison Wesley, Redwood City, CA, 1993.

[32] 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.

[33] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. In T. Yoshikawa and F. Miyazaki, editors, *Proceedings of the Third International Symposium on Experimental Robotics*, pages 501–513, Tokyo, 1993. Springer Verlag.

[34] C. L. Morgan. *Habit and instinct.* Edward Arnold, London, 1896.

[35] S. Nolfi. Using emergent modularity to develop control system for mobile robots. *Adaptive Behavior*, 5:343–364, 1997.

[36] S. Nolfi, J. L. Elman, and D. Parisi. Learning and evolution in neural networks. *Adaptive Behavior*, 3:5–28, 1994.

[37] S. Nolfi and D. Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior*, 5:75–98, 1996.

[38] 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*, pages 107–140, 1996.

[39] H. F. Osborn. Ontogenetic and phylogenetic variation. *Science*, 4:786–789, 1896.

[40] D. Parisi, F. Cecconi, and S. Nolfi. Econets: Neural networks that learn in an environment. *Network*, 1:149–168, 1990.

[41] W. A. Phillips, J. Kay, and D. Smyth. The discovery of structure by multi-stream networks of local processors with contextual guidance. *Network*, 6:225–246, 1995.

[42] E. Renshaw. *Modeling Biological Populations in Space and Time.* Cambridge University Press, Cambridge, 1991.

[43] 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.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-Propagation of Errors. *Nature*, 323:533–536, 1986.

[45] T. D. Sanger. Optimal unsupervised learning in a single-layer feedforward neural network. *Neural Networks*, 2:459–473, 1989.

[46] 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.

[47] J. Tani. Model-Based Learning for Mobile Robot Navigation from the Dynamical Systems Perspective. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:421–436, 1996.

[48] P. Turney, D. Whitley, and R. Anderson. Special Issue on Evolution, Learning, and Instincts: 100 Years of the Baldwin Effect. *Evolutionary Computation*, 4:iii–viii, 1996.

[49] C. H. Waddington. Canalization of development and the inheritance of acquired characters. *Nature*, 150:563–565, 1942.

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