Evolutionary Robots with On-line Self-Organization and Behavioral Fitness

Dario Floreano[1] and Joseba Urzelai[2]

[1]Institute of Robotic Systems (ISR)

[2]Laboratory of Microcomputing (LAMI)

Swiss Federal Institute of Technology (EPFL)

CH-1015 Lausanne, Switzerland

Abstract

We address two issues in Evolutionary Robotics, namely the genetic encoding and the performance criterion, also known as fitness function. For the first aspect, we suggest to *encode mechanisms for parameter self-organization*, instead of the parameters themselves as in conventional approaches. We argue that the suggested encoding generates systems that can solve more complex tasks and are more robust to unpredictable sources of change. We support our arguments with a set of experiments on evolutionary neural controller for physical robots and compare them to conventional encoding. In addition, we show that when also the genetic encoding is left free to evolve, artificial evolution will select to exploit mechanisms of self-organization. For the second aspect, we shall discuss the role of the performance criterion, also known as fitness function, and suggest *Fitness Space* as a framework to conceive fitness functions in Evolutionary Robotics. Fitness Space can be used as a guide to design fitness functions as well as to compare different experiments in Evolutionary Robotics.

1 Introduction

Evolutionary Robotics is a methodology for developing robotic systems that adapt to partially unknown and dynamic environments with minimal or without human intervention. The method is based on artificial evolution (Holland, 1975; Rechenberg, 1973) of a population of robots whose components (control architecture, electronic circuit, or morphology) are encoded on artificial chromosomes (bit strings). The best chromosomes, i.e. those that correspond to the most performing robots in the population, are selected for reproduction, crossed over and randomly mutated to generate a new population of chromosomes. The procedure is repeated until an individual with desired characteristics is born or until the performance of the population stops increasing (Nolfi & Floreano, 2000).

In this article we address two crucial issues in Evolutionary Robotics, namely the genetic encoding and the performance criterion, also known as fitness function. For the first aspect, we suggest to *encode mechanisms for parameter self-organization*, instead of the parameters themselves as in conventional approaches. We argue that the suggested encoding generates systems that can solve more complex tasks and are more robust to unpredictable sources of change. We support our arguments with a set of experiments on evolutionary neural controller for physical robots and compare them to conventional encoding. In addition, we show that when also the genetic encoding is left free to evolve, artificial evolution will select to exploit mechanisms of self-organization. For the second aspect, we shall discuss the role of the performance criterion, or fitness function, and suggest *Fitness Space* as a framework to conceive fitness functions in Evolutionary Robotics. Fitness Space can be used as a guide to design fitness functions as well as to compare different experiments in Evolutionary Robotics.

2 Coping with Change

The situated nature of Evolutionary Robotics is such that often evolved controllers find surprisingly simple –yet efficient– solutions that capitalize upon unexpected invariants of the interaction between the robot and its environment. For example, a robot evolved for the abil-

ity to discriminate between shapes can do so without resorting to expensive image processing techniques by simply checking the correlated activity of two receptors located in strategic positions on the retinal surface (Harvey, Husbands, & Cliff, 1994). Analogously, a robot evolved for finding a hidden location can display performances similar to those obtained by rats trained under the same conditions without resorting to complex environmental representations by using simple sensory-motor sequences that exploit geometric invariants of the environment (Lund & Miglino, 1998). The remarkable simplicity[1] and efficiency of these solutions is an advantage for fast and real-time operation required from autonomous robots, but it raises the issue of robustness when environmental conditions change. Environmental changes can be a problem also for other approaches (programming, learning, e.g.) to the extent in which the sources of change have not been considered during system design, but they are even more so for evolved systems because these rely on environmental aspects that often are not predictable by an external observer.

Environmental changes can be induced by several factors such as modifications of the sensory appearance of objects (e.g., different light conditions), changes in sensor response, rearrangement of environment configuration, transfer from simulated to physical robots, transfer across different robotic platforms, and behavioral changes of other agents in the same environment.

Some authors have suggested to improve the robustness of evolved systems by adding noise (Miglino, Lund, & Nolfi, 1996; Jakobi, 1997) and by evaluating fitness values in several different environments (Thompson, 1998). However, both techniques imply that one knows in advance what makes the evolved solution brittle in the face of future changes in order to choose a suitable type of noise and of environmental variability during evolutionary training. Another approach consists of combining evolution and learning "during life" of the individual (see (Belew & Mitchell, 1996) for a comprehensive review of the combination of evolution and learning). This strategy not only can improve the search properties of artificial evolution, but can also make the controller more robust to changes that occur faster than the evolutionary

[1] This does not imply that evolutionary approaches are restricted to forms of reactive intelligence; see for example (Floreano & Mondada, 1996).

time scale (i.e., changes that occur during the life of an individual) (Nolfi & Floreano, 1999). The combination of evolution and learning is typically achieved by evolving neural controllers that learn with an off-the-shelf algorithm, such as reinforcement learning (Ackley & Littman, 1992) or back-propagation of error (Nolfi, Miglino, & Parisi, 1994), starting from synaptic weights specified on the genetic string of the individual. Only initial synaptic weights are evolved. A limitation of this approach is the "Baldwin effect", whereby the evolutionary costs associated with learning give a selective advantage to the genetic assimilation of learned properties and consequently reduce the plasticity of the system over time (Mayley, 1996). Another limitation consists in the fact that the adaptation process is constrained by the type of learning algorithm chosen by the experimenter, which may not be the most suitable for the actual situation.

Here we suggest to *evolve the adaptive characteristics* of a controller instead of combining evolution with off-the-shelf algorithms. The method consists of encoding on the genotype a set of four local Hebb rules for each synapse, but *not the synaptic weights*, and let these synapse use these rules to adapt their weights online starting always from random values at the beginning of the life. Since the synaptic weights are not encoded on the genetic string, there cannot be genetic assimilation of abilities developed during life. In other words, these controller can rely less on genetically-inherited invariants and must develop on-the-fly the connection weights necessary to achieve the task. At the same time, the evolutionary cost of adaptation (i.e., the time and energy spent to adapt goes to the detriment of the individual's fitness) implicitly puts pressure for the generation of fast-adaptive architectures.

In preliminary investigations described in this journal (Floreano & Mondada, 1998) we compared evolution of genetically-determined weights with evolution of adaptive controllers on a simple navigation task and showed that the latter approach generates equally-good performances in less generations by taking advantage of the combined search methods. Later, we showed that evolution of adaptive controllers significantly alters the performance of robots that must cope with dynamic environments and described an experiment where co-evolutionary adaptive predators adapted on line to co-evolutionary prey robots (Floreano, Nolfi, & Mondada, 2000).

Here we describe a new set of experiments designed to further show that this approach can generate more complex controllers and test its robustness to environmental changes that were not included during evolutionary training. For what concerns adaptation to change, we focus on transfer of evolved controllers across different robotic platforms whose sensory-motor characteristics require partial re-configuration of the control system. In another set of companion papers, we also show that this approach is effective for environmental changes that involve new sensory characteristics and new spatial relationships of the environment (Urzelai & Floreano, 2000c) and in transfers from simulations to physical robots without additional evolution (Urzelai & Floreano, 2000b).

In the next sections we give an overview of the evolutionary method, describe its application to a complex sequential task, and compare the results to other encoding schemes. We then present some results on the transfer of evolved controllers across different robotic platforms. Finally, we show that when the genetic encoding itself is evolved, best individuals across generations select encoding of adaptive mechanisms rather than encoding of synaptic weights.

2.1 Encoding Mechanisms of Adaptation

The artificial chromosome encodes a set of four modification rules for each component of the neural network (components can be individual synapses or groups of synapses that converge towards the same neuron, as we shall see below), but not the synaptic strengths of the network. Whenever an artificial chromosome is decoded into a neural controller, the synaptic strengths are set to small random values. This means that the robot will initially display random actions both at the initial generation and at later generations. However, as time goes the synapses start to change their value using the genetically specified rules every 100 ms (the time necessary for a full sensory-motor loop on the physical robot). Notice that synaptic adaptation occurs on-line while the robot moves and that the network self-organizes without external supervision and reinforcement signals. The fitness function is computed along the whole duration of the robot "life". This introduces an implicit learning cost (Mayley, 1996) that gives selective advantage to individuals that can adapt faster. At the end of the life, the

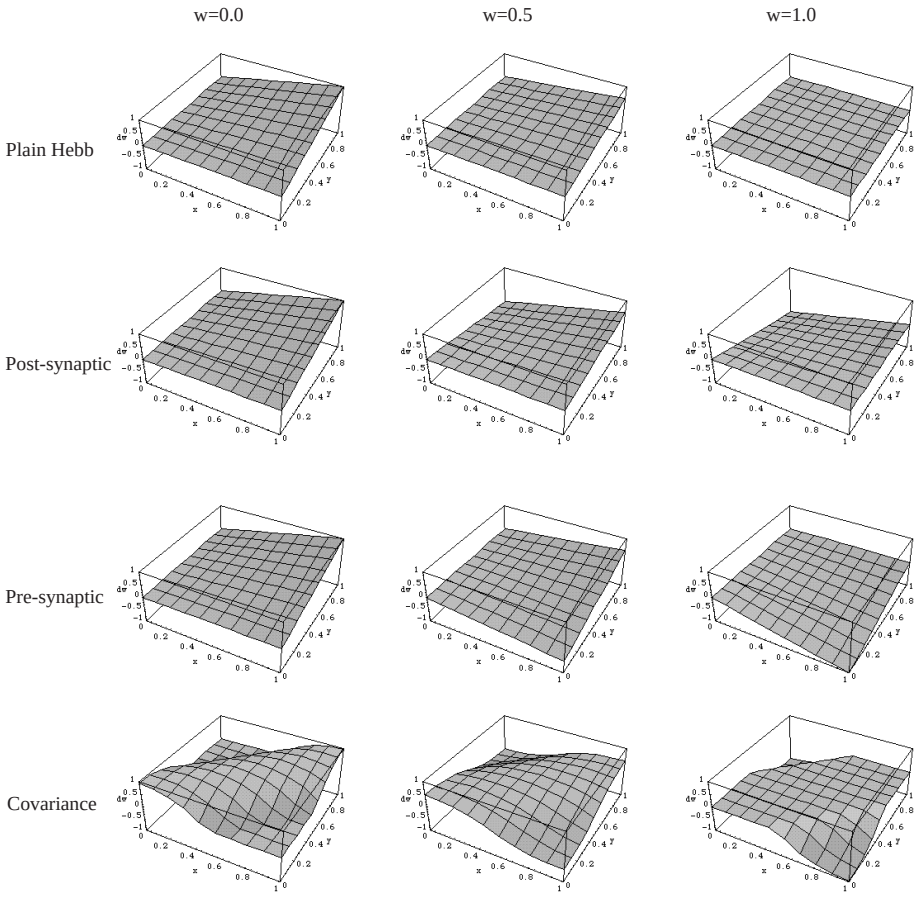

Figure 1: Synaptic change for each of the four Hebb rules. Notice that this is the *amount of change* Δw added to the synapses, not the synaptic strength. Each graph indicates the amount of change as a function of presynaptic x and postsynaptic y activity. The amount of change also depends on the current strength w of the synapse so that synapses are always bound between 0 and 1. Three graphs are shown for each rule, in the case of current strength 0.0, 0.5, and 1.0 respectively.

final synaptic strengths are not "written back" into the artificial chromosome.[2]

We have selected four types of modification rules (figure 1) to be encoded on the artificial chromosome. The choice has been based on neurophysiological findings and on computational constraints of local adaptation (the description of the rules that follows was given in a previous issue of this journal (Floreano & Mondada, 1998) and is reported here only for sake of clarity). In other words, these rules capture some of the most common mechanisms of local synaptic

[2]In other words, we use Darwinian evolution instead of Lamarckian evolution where the effects of learning are encoded in the artificial chromosome. See (Yamamoto, Sasaki, & Tokoro, 1999) for an experimental comparison between these two types of evolution in changing environments.

adaptation found in the nervous systems of mammalians (Willshaw & Dayan, 1990). These rules were modified in order to satisfy the following constraints. Synaptic strength could not grow indefinitely, but was kept in the range $[0, 1]$ by means of a self-limiting mechanism which depended on synaptic strength. Because of this self-limiting factor, a synapse could not change sign, which was genetically specified, but only strength. Each synaptic weight w_{ij} is randomly initialized at the beginning of the individual's life and is updated after every sensory-motor cycle (100 ms),

$$w_{ij}^{t} = w_{ij}^{t-1} + \eta \Delta w_{ij},$$

where $0.0 < \eta < 1.0$ is the learning rate and Δw_{ij} is one of the four modification rules specified in the genotype:[3]

1. *Plain Hebb rule*: can only strengthen the synapse proportionally to the correlated activity of the pre- and post-synaptic neurons.

$$\Delta w = (1 - w)\, xy \tag{1}$$

2. *Postsynaptic rule*: behaves as the plain Hebb rule, but in addition it weakens the synapse when the postsynaptic node is active but the presynaptic is not.

$$\Delta w = w\,(-1 + x)\, y + (1 - w)\, xy \tag{2}$$

3. *Presynaptic rule*: weakening occurs when the presynaptic unit is active but the postsynaptic is not.

$$\Delta w = wx\,(-1 + y) + (1 - w)\, xy \tag{3}$$

4. *Covariance rule*: strengthens the synapse whenever the difference between the activations of the two neurons is less than half their maximum activity, otherwise the synapse is weakened. In other words, this rule makes the synapse stronger when the two neurons

[3]These four rules co-exist within the same network.

have similar activity and makes it weaker otherwise.

$$\Delta w = \begin{cases} (1-w)\mathcal{F}(x,y) & \text{if } \mathcal{F}(x,y) > 0 \\ (w)\mathcal{F}(x,y) & \text{otherwise} \end{cases} \tag{4}$$

where $\mathcal{F}(x,y) = \tanh(4(1-|x-y|)-2)$ is a measure of the difference between the presynaptic and postsynaptic activity. $\mathcal{F}(x,y) > 0$ if the difference is bigger or equal to 0.5 (half the maximum node activation) and $\mathcal{F}(x,y) < 0$ if the difference is smaller than 0.5.

The genetic encoding refers to the way in which the neural controller is mapped into a bit string that represents the artificial chromosome of an individual. A gene is a set of bits that encode a given feature of the neural controller. We consider two aspects of genetic encoding: the *feature* level and the *properties* of that feature.

Features. We consider two levels of feature encoding that reflect the two basic components of a neural networks: the synapses and the nodes (see top of figure 2). *Synapse Encoding* refers to the case where a gene encodes the properties of an individual synapse. In this case, the artificial chromosome will have as many genes as synapses in the network. This type of encoding is rather common in works that combine evolutionary computation and neural networks; it is also known as "direct encoding" (Yao, 1993). *Node Encoding* instead refers to the case where a gene encodes the properties of an individual node. In that case, all the incoming synapses to that node will share the same properties specified for that node (except for the sign of the traversing signal which is a property of the presynaptic node). The artificial chromosome will have as many genes as nodes in the network. Synapse Encoding allows a detailed definition of the neural network, but for a fully connected network of N neurons the genetic length is proportional to N^2. Instead Node Encoding requires a much shorter genetic length (proportional to N), but it allows only a rough definition of the controller.

Properties. Irrespectively of the feature level chosen (synapses or nodes), each gene is composed of five bits that represent the properties of the corresponding feature. We consider

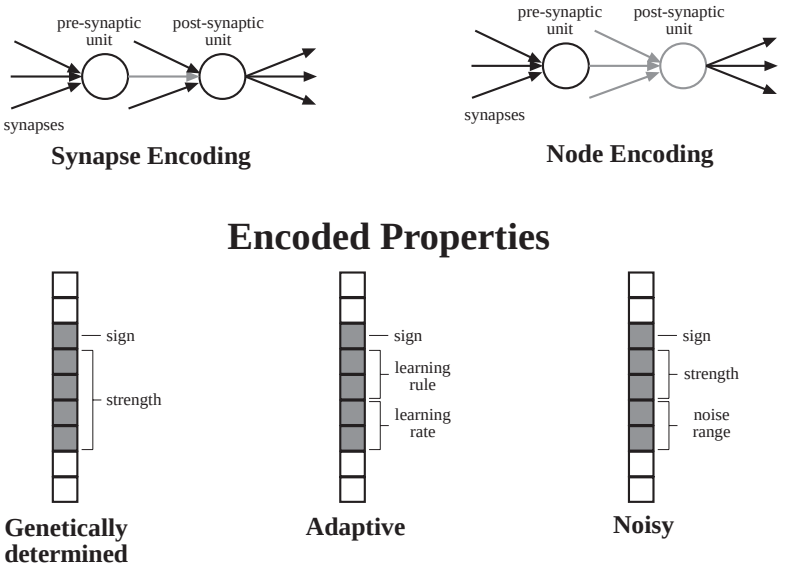

Figure 2: Different type of Genetic Encoding. **Top**: Two feature levels. The genetic string can either encode the properties of each individual synapse in the network (Synapse Encoding) or encode the properties of each node only (Node Encoding). In the latter case, the encoded properties are applied to all incoming synapses to that node. Node Encoding results in shorter genetic strings. **Bottom**: Three types of properties. Genetically-determined properties specify the connection sign and strengths of synapses. Adaptive properties specify the sign, the adaptation rule, and the learning rule of the synapses. Noisy properties specify the sign, weight strength, and a noise range that is continuously applied to the synapse. Properties are applicable to both Synapse and Node encoding, but in the latter case all incoming synapses will be have the same properties.

three types of properties (see bottom of figure 2 and the table). For all three types, the first bit always represents the sign of the signal travelling outward (either through the synapse in the case of *synapse encoding* or through the outgoing axon in the case of *node encoding*). The remaining four bits can encode the following properties:

Encoding	Bits for one synapse / node				
Genotype	1	2	3	4	5
A	sign	strength			
B	sign	Hebb rule		rate	
C	sign	strength		noise	

Table 1: Genetic encoding of synaptic parameters for Synapse Encoding and Node Encoding. In the latter case the sign encoded on the first bit is applied to all outgoing synapses whereas the properties encoded on the remaining four bits are applied to all incoming synapses. A: Genetically determined controllers; B: Adaptive synapse controllers; C: Noisy synapse controllers.

1. *Genetically-determined synapses*: 4 bits encode the synaptic strength. This value is constant during "life" of the individual. This is the conventional way of evolving neural networks (Yao, 1993).

2. *Adaptive synapses*: 2 bits encode the 4 adaptive rules described above and 2 bits the corresponding learning rate. Synaptic weights are always randomly initialized at the beginning of an individual's life and then updated on-line every 100 ms according to their own adaptation rule while the individual interacts with the environment. This is the core of the methodology proposed in this paper, that is evolving the mechanisms of self-organization of a controller.

3. *Noisy synapses*: 2 bits encode the weight strength and 2 bits a noise range. The synaptic strength is genetically determined at birth, but a random value extracted from the noise range is freshly computed and added every 100 ms while the individual interacts with the environment. This is a control condition to check whether the effects of random variations are equal or different from the mechanisms of self-organization described above.

In previous work (Floreano & Mondada, 1998) we used only Synapse Encoding and showed that evolution of Adaptive synapses produces in less generation better controllers than evolution of Genetically-determined synapses for simple reactive navigation. In this paper we wish to go one step further and investigate whether Adaptive Synapses can use Node Encoding which is a much more compact representation. In the next set of experiments, we will compare evolution of Adaptive synapses with Node Encoding to other types of genetic encoding for a non-trivial sequential task.

2.2 A Sequential Task: The "Light-Switching" Problem

A mobile robot Khepera equipped with a vision module is positioned in the rectangular environment shown in figure 3. A light bulb is attached on one side of the environment. This light is normally off, but it can be switched on when the robot passes over a black-painted area on the opposite side of the environment. A black stripe is painted on the wall over the light-switch area. Each individual of the population is tested on the same robot, one at a time,

Figure 3: A mobile robot Khepera equipped with a vision module gains fitness by staying on the gray area only when the light is on. The light is normally off, but it can be switched on if the robot passes over the black area positioned on the other side of the arena. The robot can detect ambient light and the color of the wall, but not the color of the floor.

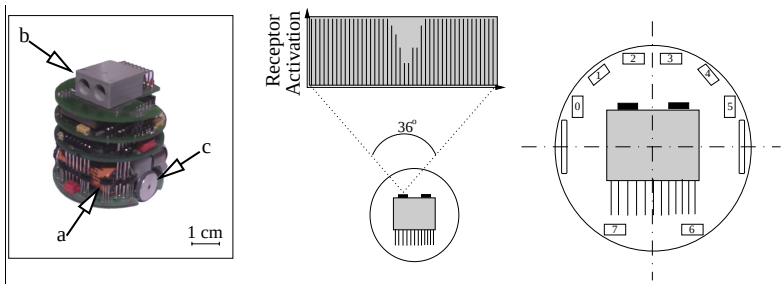

Figure 4: The Khepera robot used in the experiments. Infrared sensors (a) measure object proximity and light intensity. The linear vision module (b) is composed of 64 photoreceptors covering a visual field of 36° (center). The output of the controller generates the motor commands (c) for the robot. Right figure shows the sensory disposition of the Khepera robot.

for 500 sensory motor cycles, each cycle lasting 100 ms. At the beginning of an individual's life, the robot is positioned at a random position and orientation and the light is off.

The fitness function is given by the number of sensory motor cycles spent by the robot on the gray area beneath the light bulb *when the light is on* divided by the total number of cycles available (500). In order to maximize this fitness function, the robot should find the light-switch area, go there in order to switch the light on, and then move towards the light as soon as possible, and stand on the gray area. Since this sequence of actions takes time (several sensory motor cycles), the fitness of a robot will never be 1.0. Also, a robot that cannot manage to complete the entire sequence will be scored with 0.0 fitness. A light sensor placed under the robot is used to detect the color of the floor—white, gray, or black— and passed to a host computer in order to switch on the light bulb and compute fitness values. The

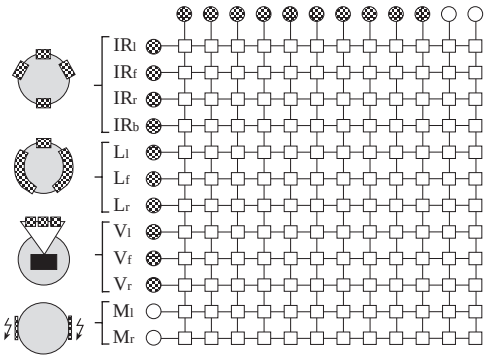

Figure 5: The neural controller is a fully-recurrent discrete-time neural network composed of 12 neurons giving a total of 12 x 12= 144 synapses (here represented as small squares of the unfolded network). 10 sensory neurons receive additional input from one corresponding pool of sensors positioned around the body of the robot shown on the left (l=left; r=right; f=front; b=back). $\vec{IR}$=Infrared Proximity sensors; $\vec{L}$=Ambient Light sensors; $\vec{V}$=vision photoreceptors. Two motor neurons $\vec{M}$ do not receive sensory input; their activation sets the speed of the wheels ($M_i > 0.5$ forward rotation; $M_i < 0.5$ backward rotation)

output of this sensor is *not* given as input to the neural controller. After 500 sensory motor cycles, the light is switched off and the robot is repositioned by applying random speeds to the wheels for 5 seconds.

Notice that the fitness function does not explicitly reward this sequence of actions (which is based on our external perspective), but only the final outcome of the sequence of behaviors chosen by the robot. We shall come back to the issue of fitness design later on in section 3. The controller we have used in our experiments is a fully-recurrent discrete-time neural network. It has access to three types of sensory information from the robot (figures 4 and 5):

1. *Infrared light*: the active infrared sensors positioned around the robot (figure 4, a) measure the distance from objects. Their values are pooled into four pairs and the average reading of each pair is passed to a corresponding neuron.

2. *Ambient light*: the same sensors are used to measure ambient light too. These readings are pooled into three groups and the average values are passed to the corresponding three light neurons.

3. *Vision*: the vision module (figure 4, b) consists of an array of 64 photoreceptors covering a visual field of 36° (figure 4, center). The visual field is divided up in three sectors and

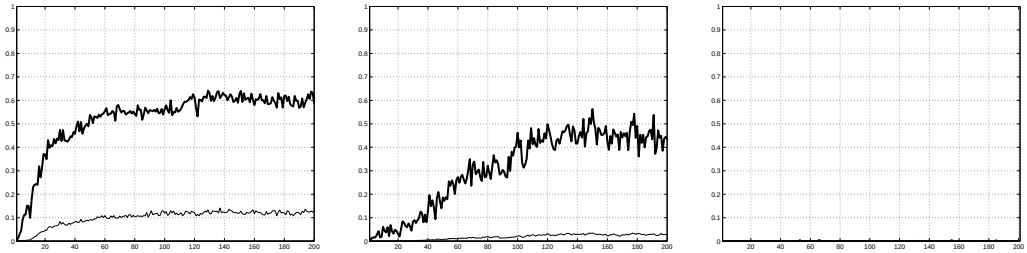

Figure 6: Fitness values across generations of controllers evolved with Adaptive synapses and Node Encoding (*left*), Genetically-determined synapses and Synapse Encoding (*center*), and Genetically-determined synapses and Node Encoding (*right*). The last graph looks empty because the fitness values are always zero. Thick line=best individual; thin line=population average. Each data point is an average over 10 replications with different random initializations. Population size is 100 and 20 best individuals reproduce by making 5 copies. Crossover probability is 0.2 and mutation probability is 0.05 (per bit).

the average value of the photoreceptors (256 gray levels) within each sector is passed to the corresponding vision neuron.

Two motor neurons are used to set the rotation speeds of the wheels (figure 4, c). Neurons are updated every 100 ms.

The fitness data recorded during evolution and reported in figure 6 show two main results. The first is that Genetically-determined synapses with Node encoding (graph on the right) report zero fitness. This is due to the fact that all incoming synapses to a node have always the same strength and cannot change. Therefore, Adaptive Synapses with Node Encoding (graph on the left) must be compared to Genetically-determined synapses with Synapse Encoding (graph on the center). The second result is that evolved individuals with Adaptive Synapses report higher fitness values (0.6 against 0.5) and reach the best value obtained by Genetically-determined individuals in less than half generations (40 against more than 100). In the condition of Noisy synapses with Synapse Encoding and Node Encoding, all best individuals report fitness values in the range 0.1-0.2 (data not shown). This result indicates that the changes induced by the adaptive rules are not equivalent to some random changes.

Figure 7 shows the behaviors of two best individuals evolved with adaptive synapses and Node Encoding (left), and with genetically-determined weights and Synapse Encoding (right).

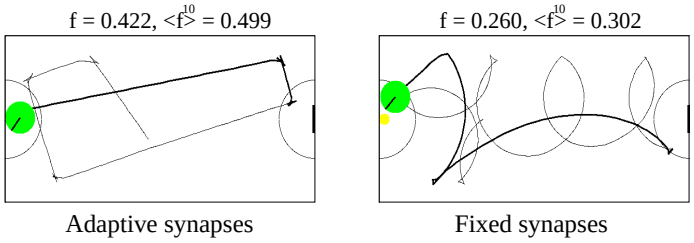

Figure 7: Behaviors of two best individuals (from last generation) with adaptive synapses and Node Encoding (*left*) and with genetically-determined synapses and Synapse Encoding (*right*). When the light is turned on, the trajectory line becomes thick. The corresponding fitness value is printed on the top of each box along with the average fitness of the same individual tested ten times from different positions and orientations.

In both cases individuals aim at the area with the light switch[4] and, once the light is turned on, they move towards the light and remain there. The better fitness of the adaptive controllers (given on the top of each box, see figure caption) is given by straight and faster trajectories showing a clear behavioral change between the first phase where they go towards the switching area and the second phase where they become attracted by the light. Instead, genetically-determined individuals display always the same looping trajectories around the environment with some attraction towards the stripe and the light. This minimalist behavior, which depends on invariant geometrical relations of the environment, gives them a chance to accomplish the task but with a lower performance.

Additional behavioral tests and network analyses clearly indicated that evolved adaptive individuals achieve higher fitness because they rapidly modify their weights in ways that reflect the subtask at hand and functionally related to the survival criterion (Floreano & Urzelai, 1999; Urzelai & Floreano, 2000a).

2.3 Cross-platform Adaptation

Cross-platform transfer is a very useful feature, but we are not aware of any control system that can be transferred across different robots without changes. Cross-platform becomes useful in adaptive and evolutionary systems where initial training experiences can cause the robot

[4]Their performance is badly affected if the vision input is disabled, indicating that they do not use random search to locate the switch (data not shown).

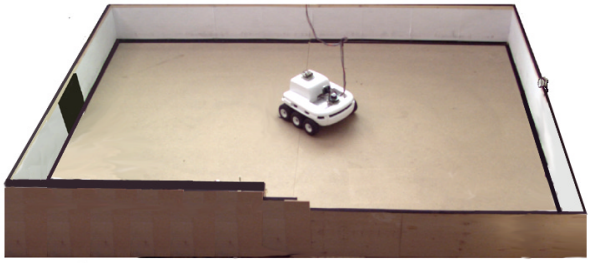

Figure 8: A mobile robot Koala equipped with a vision module gains fitness by staying near the lamp (right side) only when the light is on. The light is normally off, but it can be switched on if the robot passes near the black stripe (left side) positioned on the other side of the arena. Position of the robot is controlled by an external positioning system and passed to the computer in order to control the light and to compute the fitness.

to produce harmful actions. One may train (or evolve) control systems for a desktop sturdy robot like the miniature Khepera and then download them to larger and consequently more fragile robots[5]. In this case, it would be desirable that the control system self-adapts to the new sensory-motor characteristics and morphology.

In previous work we have shown that this can be achieved by using incremental evolution of genetically-determined networks (Floreano & Mondada, 1998). However, even for a simple reactive navigation behavior it took additional 20 generations to re-adapt to the new robot. Here we test the adaptive properties of the evolutionary adaptive strategy by transferring onto a physical Koala robot (figure 8) the best individuals of the last generation evolved on the miniature Khepera robot. A mobile robot Koala equipped with a vision module is positioned in the rectangular environment shown in figure 8. As in the previous experiment with the Khepera robot, the Koala robot must find the light-switching area, go there in order to switch the light on, and then move towards the light as soon as possible and stay there in order to score fitness points.

The Koala robot has six wheels driven by two motors (one on each side) and 16 infrared sensors (figure 9) with a different and stronger detection range than those used on the Khepera robot. An external positioning system emitting laser beams at predefined angles and frequencies is positioned on the top of the environment and the Koala robot is equipped with an

[5]Obviously, the two robots must share some characteristics, such as type of sensors and actuators used, that allow a suitable interfacing of the control system.

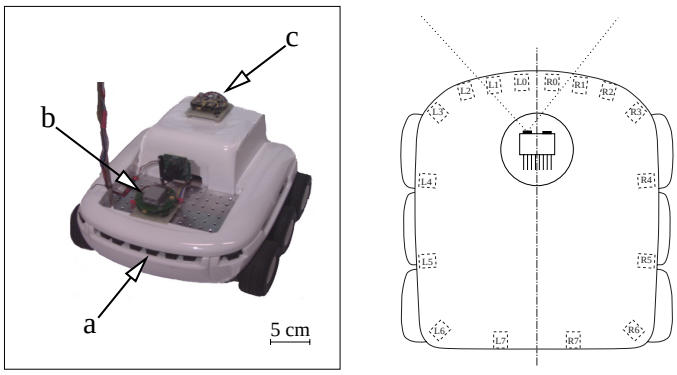

Figure 9: The Koala robot used in the experiments. Infrared sensors (a) measure object proximity and light intensity. The linear vision module (b) is the same as used in the experiments with the Khepera robot. The localization module (c) provides the position of the robot at every time step. Right figure shows the sensory layout of the Koala robot. Only 8 equally-spaced sensors are selected as input to the network.

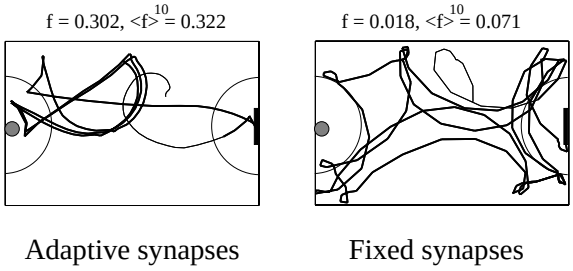

Figure 10: Behaviors of individuals with adaptive synapses (left) and genetically-determined synapses (right) tested on the Koala robot. Individuals belong to the last generation evolved in simulation for the Khepera robot.

additional turret capable of detecting laser and computing in real-time the robot displacement. This information is used in order to control the light and to compute the fitness. The performance of adaptive individuals is not affected by the transfer from the Khepera robot to the Koala robot whereas genetically-determined individuals report a significative fitness loss (for more details see (Urzelai & Floreano, 2000b)).

Adaptive individuals correctly approach the light-switching area and are clearly attracted by light (figure 10, left). As in the case of real Khepera robot, once arrived under the light the Koala robot moves around the fitness area while remaining close to it until the testing time is over. On the other hand, genetically-determined individuals (right) perform spiralling trajectories around the environment and do not display any attraction by the black stripe or

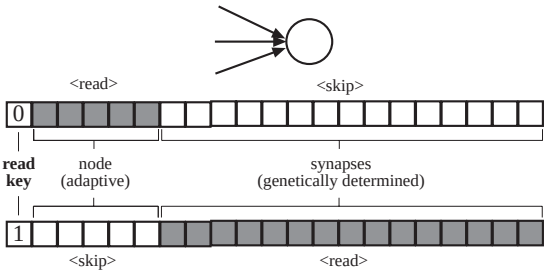

Figure 11: Genetic representation used for the experiments on evolution of encoding. The figure shows a piece of chromosome that stands for a node with three incoming synapses. The genetic representation consists of three parts: one bit that tells how to read the remaining parts, 5 bits that encode the adaptive properties of the node, and 5*3 bits that encode the weight strengths for each of the three sysnapses. If the first bit is zero, only the node properties are read and the rest is skipped; otherwise the node properties are skipped and all the remaining weight strengths of the synapses are read and mapped into the neural controller.

the light. They eventually manage to pass through the light-switching area, turn the light on, and occasionally score some fitness points passing over the fitness area. In several cases, genetically-determined individuals get stuck against the walls of the environment (behaviors not shown).

2.4 Evolution of Genetic Encoding

In the experiments described above we have compared different encoding strategies. In other words, for each experimental condition the system was constrained to use only a predefined encoding strategy, but it was not possible to tell whether a combination of different encoding strategies may have generated better results, and what type of combination would that be. In this section we investigate these questions by encoding adaptive and genetically-determined strategies and evolving what parts of the genetic string are read and mapped into the neural controller. In particular, we are interested in evolving combinations of Synapse encoding of genetically-determined synapses with Node encoding of adaptive synapses, which respectively represent the conventional approach to neural evolution and the approach that reported best results in the experiments described above.[6]

[6]Notice that comparing genetically-determined and adaptive synapses both with Node encoding would not be a fair comparison because we have already shown that the genetically-determined synapses with Node encoding report zero fitness (rightmost graph of figure 6).

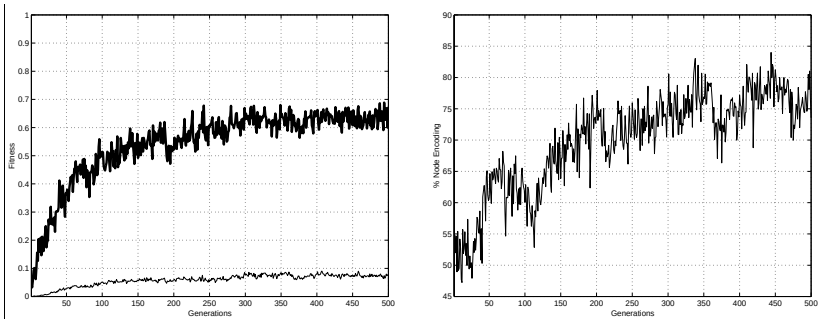

Figure 12: Experimental results for evolution of encoding. *Left*: Best (thick line) and average (thin line) fitness across generations. *Right*: Percentage of nodes in the networks of the best individuals that use node encoding of adaptive properties. Each data point is an average over 10 replications with different random initializations. Population size is 100 and 20 best individuals reproduce by making 5 copies. Crossover probability is 0.2 and mutation probability is 0.05 (per bit).

For each node in the network (figure 11), the artificial chromosome encodes both the adaptive properties of the node and the genetically-determined weight strengths of all incoming synapses to that node. These two encoding strategies are identical to those already described in the previous sections. In addition, a single bit is added to the beginning of the genetic string for each node. If that bit value is off, only the node properties are read and the rest of the genes is skipped. In this case, the incoming synapses of that node will be randomly initialized and will be updated using the node adaptive rules. Instead, if the bit value is on, the node properties are skipped and the remaining bits for that node specifying the weight strengths of each synapse are read. In this case, the incoming synapses of that node will be assigned genetically-determined strengths and won't be updated during the life of the individual.

The experimental results indicate that individuals evolved under this condition reach the same fitness levels (figure 12, left) reported in the case of evolution of node encoding (figure 6, left). This result answers our first question, that is node encoding of adaptive mechanisms is not worse than a combination of different encoding strategies. Furthermore, the percentage of nodes that resort to node encoding of adaptive rules in the networks of the best individuals increases across generations and remains significantly higher than the percentage of nodes that resort to synapse encoding of genetically-determined weights (figure 12, right). In other words, when evolution can select what type of genetic encoding is used to generate the neural controller, it tends to select node encoding of adaptive rules.

2.5 Discussion

The results described in this paper show that genetic encoding of mechanisms of adaptation supports evolution of more performant controllers than genetic encoding of the parameters themeselves. Furthermore, cross-platform tests indicate that these evolved individuals can perform the same task in conditions that were not experienced during evolution.[7] Instead, individuals evolved using genetic encoding of the synaptic weights become tuned to the evolutionary conditions and do not perform well in new environments. Finally, we have shown that when the genetic encoding is evolved, artificial evolution systematically selects encoding of adaptive mechanisms.

Evolution of learning rules for neural networks has been investigated also by Chalmers (Chalmers, 1991) in the framework of supervised learning. Chalmers encoded in the artificial chromosome the coefficients of a polynomial function of synaptic variables and applied the corresponding learning rule to all the weights of a neural network trained on a set of pattern classification problems. He showed that artificial evolution often re-discovered the Delta rule (Widrow & Hoff, 1999). The main differences between his approach and ours—beside differences in the application domain—are that in our case: a) we are not attempting to optimize rules for specific learning modalities and architectures, but consider encoding of rules as a general way to improve the dynamics of the controller; b) the same network can host a variety of different local adaptation mechanisms whose combined effect may (or may not) produce the appearance of a learning behavior; c) we investigate different encoding strategies, compare the results with conventional techniques, and also investigate evolution of the genetic encoding itself; d) the fitness function is based on the behavior of the system along the whole "life duration" (whereas in Chalmers' case the fitness was based on the performance of the networks after training), which puts intrinsic selection pressure for architectures that self-organize online and quickly in order to reduce adaptation costs. We shall discuss the choice of fitness functions in more detail in the next section.

[7] In other papers, we show evidence for such robustness also in the case of changes due to sensory stimulation and to transfer from simulations to physical robots (Urzelai & Floreano, 2000c), and in the case of re-arrangements of the geometrical layout of the environment (Urzelai & Floreano, 2000b).

Since adaptive encoding can operate at the level of node encoding, this strategy may become useful for artificial evolution of network morphology. Some methods already exist for evolution of network architectures ((Kitano, 1990; Gruau, 1994; Nolfi & Parisi, 1995), for example), but these strategies often need quite complex specifications of node types to ensure diverse connectivity, require much domain-specific knowledge (such as whether symmetries exist), and have not yet been shown to be competitive with conventional synapse-encoding methods. Instead, evolution of adaptive mechanisms and node encoding is a simple and compact schema that leaves the specification of synaptic strengths to the adaptive mechanisms and may potentially lead to specification of complex architectures. Initial investigations in this direction show that this encoding strategy scales up very well for very large network architecture (Floreano & Urzelai, 1999).

The reader may have noticed that we have never used the term *learning* to describe the behavior of individuals with changing synapses.[8] The reason is that we have no evidence that evolved individuals are effectively learning something in the cognitive sense of acquiring new competences and storing knowledge. Instead, we used the term "self-organization" and "adaptation" to indicate that the modifications induced by the genetically evolved combination of rules produce behavioral changes that are functionally related to the fitness criterion. In another current project related to the evolution of morphologies described above, we are looking at ways in which architectural constraints (e.g., synapse-to-synapse mechanisms (Körding, K. and König, P., 2000) and lateral modulatory connections (Kay, Floreano, & Phillips, 1997)) may induce incremental and long-lasting modifications that may be described as learning.

3 Fitness Space

The fitness function decides which individuals are selected for reproduction. If there is no fitness function and individuals are randomly selected, the effects of reproduction amount to genetic drift.

[8] Except when we referred to the Hebbian *learning* rules and to the *learning* rate η to be consistent with most of the connectionist literature.

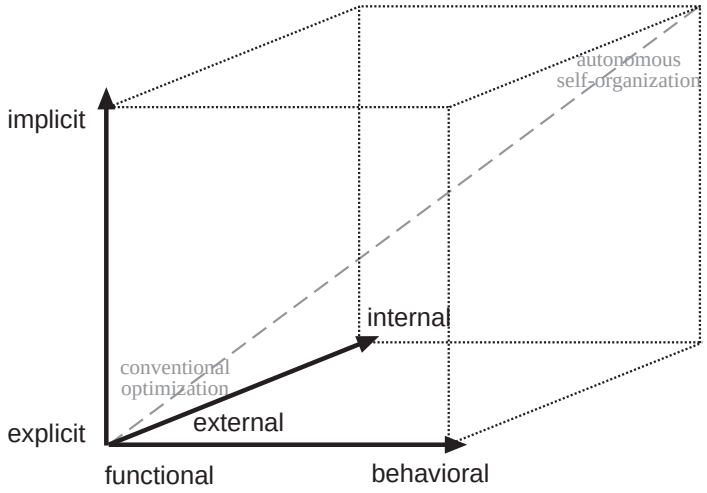

Figure 13: Fitness Space is a framework for describing and comparing fitness functions along three dimensions. It provides a nominal and ordinal scaling of functions. The diagonal between the lower-left corner and the upper-right corner defines a continuum of functions between conventional optimization approaches and generation of autonomous self-organizing systems.

In Evolutionary Robotics, the choice of fitness function has strong consequences on the applicability of the approach to physical robots, evolvability of the robot, dynamics of the evolutionary process, and ultimately on the outcome of the evolutionary process. Most people struggle with the choice of suitable functions using a trial-and-error strategy. People using artificial evolution for Automated Engineering are most affected by this process because they often have precise expectations about the desired result. Unfortunately, there is not a way to infer a fitness function from a description of the expected result. Typically, one comes up with a function based on one's own experience, tries it out, and then gradually modifies it to accomodate additional constraints. Although conceiving a fitness function suitable for a desired behavior is often easier than designing the corresponding program (Dorigo & Colombetti, 1998), the widespreading use of Evolutionary Robotics requires a better understanding of the decisions involved in setting up a fitness function.

In this section we propose *Fitness Space* as a framework to describe, assess, and compare fitness functions. Fitness Space can be used as a guideline to choose among alternative fitness functions according to one's goals, but it does not provide a recipe to design specific fitness

functions for specific problems. Fitness Space is defined by three dimensions[9] (figure 13).

3.1 Functional-Behavioral Dimension

The first dimension is given by the continuum between Functional and Behavioral fitness. A purely functional fitness is based only on components that directly measure the way in which the system functions. For example, in an early attempt to evolve a neural controller for a walking robot, Lewis *et al.* used a functional fitness that measured the frequency and amplitude of the oscillations of the evolutionary controller (Lewis, Fagg, & Solidum, 1996). The closer these two components were to the desired pattern, the higher the fitness of the individual.[10] This implies that the authors knew what type of oscillatory dynamics were required for producing a certain behavior. On the other hand, a purely behavioral fitness is based only on components that measure the behavior of the individual's behavior. To stay with the example of the walking robot, a behavioral function would be proportional to the distance covered by the robot in a given amount of time. Another way of describing the difference between these two fitness extremes is that functional fitness evaluates the *causes* of behavior whereas behavioral fitness evaluate the *effects* of behavior. Either choice has strong implications. A functional fitness can ensure the highest match between evolved and desired behavior, but it is quickly compromised as soon as the functional components of the controller do not match any longer mechanical or environmental factors. For example, a wheel may gradually wear out and induce a direction bias in the trajectory of the robot that would go undetected by a purely functional fitness function. Instead, behavioral fitness can produce viable results even in the presence of some mechanical and sensory defects because the control system will implicitly accommodate them, but the results may not match the expectations. For example, the evolved robot may crawl instead of walk. The position of a given fitness function along the Functional-Behavioral dimension depends on the number of these two types of components and their relative weights.

[9]Fitness Space should not be confused with Fitness Landscape which instead describes the distribution of fitness values for all possible combinations of the bits that compose the artificial chromosome.

[10]Functional fitness was used mainly in an early stage of evolution. In later stages, the authors added further behavioral components to the fitness.

The fitness function used in the experiments described in this paper is completely behavioral because it is based on the effects of the robot behavior, that is the amount of time spent under the light when this is on.

3.2 External-Internal Dimension

The dimension along the External-Internal continuum refers to availability—with respect to the robot—of the variables that compose the function. An external fitness component is one that cannot be measured directly by the robot. For example, the exact distance between a robot and an obstacle can be measured only by external positioning devices or by an external observer.[11] An internal fitness component instead is one that can be measured by the robot itself, such as the energy level or the state of its own sensors. The difference is subtle, but very important for Evolutionary Robotics. For example, external fitness functions are often used in software simulations of robots where all aspects of the system are directly available to the programmer. Here the distinction between internal and external variables is only formal because both types of variables are readily available. External fitness functions are popular because they allow a detailed and precise assessment of the robot performance and also because they naturally conform to the perspective of an external observer (in other words, it is easier to design a function that fits one's perspective of the expected behavior). However, real robots cannot always be evolved using external fitness components and, when feasible, this implies resorting to *ad hoc* and expensive devices, such as a Global Positioning System or an external camera with tracking software. Therefore, the choice of an external function should be carefully evaluated when devising an evolutionary robotic system. For example, external fitness functions are not recommended if one decides to start evolving the robot in simulation and later wishes to incrementally carry on evolution on a physical robot. Similarly, one should be careful about making strong claims on results obtained using external fitness functions because these results might not be easily applicable to many real-world situations where the necessary measuring devices are not available.

[11] Notice that distances cannot be reliably estimated by odometry and proximity sensors on the robot (such as sonar and active infrared).

In the context of autonomous robots, we think that external fitness functions do not give a higher probability of obtaining desired results than internal fitness functions because they are based on the perspective of an external observer. In other words, the robot may be forced to display a behavior that is too difficult to implement given the characteristics of its sensory-motor apparatus or of its control architecture. Resorting to internal fitness function thus has the additional advantage of forcing the engineer not to rely too much on assumptions derived from an external and potentially inaccurate perspective.

The fitness function used in the experiments of this paper was internal because the light sensors around and underneath the platform were used to detect whether the robot was on the gray area and whether the light was on. In other words, the system was self-contained because it did not require external devices to assess its states and performance.

3.3 Explicit-Implicit Dimension

The Explicit-Implicit dimension refers to the quantity of constraints explicitly imposed by a human person to select individuals for reproduction. An approximate indicator is given by the number of components included in the fitness function. The higher the number of components, the more explicit the fitness function is.

Artificial Life approaches (Langton, 1990) aimed at studying evolution of ecosystems tend to use implicit fitness functions in order to ensure ecological validity because in real life there is not an explicit fitness function. For example, artificial organisms may reproduce only if and when their energy levels reach a certain threshold (these type of ecosystems are also known as *Latent Energy Environments* (Menczer & Belew, 1993)). Compare this with a situation where the fitness function explicitly rewards—for example—the quantity of food items gathered, distance from predators, and the ability to recognize conspecifics.

Automated Engineering approaches (Dorigo & Colombetti, 1998) instead tend to resort to more explicit fitness functions in the attempt to actively steer the evolutionary system towards desired behaviors. This may sound reasonable, but in practice it can get out of control very quickly. As the number of constraints increases, one is faced with the problem of how to weight and combine them (addition, product, e.g.). Furthermore, a higher number of components can

increase the probability of local minima and make the problem too hard for an initial random population (bootstrap problem). Once again, these problems can be partly explained by the fact that fitness constraints are chosen from an external perspective and thus they may not be fully matched by the robotic hardware and control architecture.

We think that explicit fitness functions are in contrast with the search of emergent forms of artificial intelligence because although the resulting evolved systems have not been pre-programmed their abilities have largely been decided and constrained by an external observer. Such evolved systems can hardly display unexpected abilities and under some definitions they may not be called emergent (for example, when emergence is defined as the degree of surprise (Ronald, Sipper, & Capcarrère, 1999)).

In the experiments described in this paper we have tried to keep the contraints as low as possible by simply accumulating one fitness point for every sensorymotor loop spent over the gray area when the light was on, but not rewarding the robot for abilities such as wall avoidance, straight trajectories, visual discrimination, light following, etc.

3.4 Comparing Evolutionary Experiments

Among the four methods to compare, or scale, experimental observations (Torgerson, 1958), Fitness Space supports only categorical and ordinal scales.[12] For example, the diagonal between the lower-left corner and the upper-right corner defines a continuum between two categories: "conventional optimization" approaches and "self-organization of autonomous systems" (figure 13). Although the point of separation between the two approaches is categories, we can say that experiments falling in the lower-left region are concerned with automatic engineering whereas experiments falling in the upper-right region are concerned with emergent autonomous systems.

We can also order two experiments according to the components of their fitness functions.

[12] *Categorical, also known as nominal, scales*, the most primitive method, group observations into qualitative classes. For example, one may classify approaches in robotics as "bio-inspired", "adaptive", "cartesian", etc. *Ordinal scales* are more informative and can be used only when one can assign to each observation a number that reflects some quantitative property. Ordinal scale can tell only whether there is a difference between two observations and in what direction the difference goes. For example, we may say that one mineral is softer than another because it is damaged when they are scratched together. However, with ordinal scales we do not know the true magnitude of oberved phenomena.

To go back to our example of legged robots, consider two imaginary experiments aimed at evolving walking controllers for these robots. Imagine that the components used in the fitness functions are:

a=oscillation frequency of leg controller (functional, internal);

b=distance covered (behavioral, external);

c=state of motors (behavioral, internal);

d=state of bump sensor under the belly (behavioral, internal);

2=constant (affects relative position along implicit/explicit dimension).

Fitness function f^1

$$f^1 = (2 * a) + (b * d) \tag{5}$$

is composed of two additive parts. The first part rewards the controller for producing predetermined oscillation frequencies that correspond to a desired motion pattern for each leg. This part has a strong weight (factor 2). The second part instead adds to it rewards for the distance covered by the robot body multiplied by the state of the sensors. In other words, robots that move longer without creeping over the floor get higher fitness.

Fitness function f^2

$$f^2 = c + d \tag{6}$$

has two parts too. The first part is maximized by the quantity of current sent to the motors and the second is maximized when the robot does not touch the floor. In other words, robots that keep their legs moving but do not stay on the floor will receive higher fitness.

Function f^1 therefore is *less implicit, less internal, and less behavioral* than function f^2. Assuming that all other conditions are the same, fitness f^1 may generate efficient and specific gaits (depending on the values of a), but it may take more generations and a larger population because the number of constraints are likely to make the fitness lansdcape very hard. Also, it requires the use of additional hardware to measure the distance covered by the robot. An experiment of this type falls in the category of "automated engineering" and its validity should be assessed accordingly.

Fitness f^2 instead is not guaranteed to generate efficient gaits, in fact it may well generate

robots that dance without covering much forward distance. However, the evolved solutions will be more dependent on the interactions between the robot and its environment. Also, this function does not require additional devices and is portable to a range of different robots with unknown dynamics and kinematics because it does not require functional knowledge of the system. This type of experiment is closer to the category of "emergent self-organization". We believe that Fitness Space can help to compare apparently similar experiments and better assess the significance of their results.

As a final note, it should be noticed that the fitness function is not the only factor that determines the outcomes and evolvability of a system. Although it plays an important role, other determinant factors include the type of sensors used, the environmental setup, the type of challenge that is being addressed, and various aspects of the controller architecture and genetic encoding. Although our analysis is valid only when all these factors are kept constant, we believe that it provides a more general framework for reading through the growing literature in Evolutionary Robotics.

4 Conclusions

In this paper we have addressed two key aspects of artificial evolution in robotics, namely the genetic encoding and the fitness function, and described an experiment that summarizes our approach to Evolutionary Robotics. This approach consists of evolving controllers for physical robots that display intrinsic adaptation abilities and are based on behavioral and internal fitness functions.

We believe that these are two important features for the next generation of robotic applications, that is *personal and service robotics*. Personal robots are machines that interact with humans and are used for entertainment, education, and assistance. Service robots are machines used to carry out work in unstructured environments, such as demining, cleaning, space exploration. Personal and service robots entice an entirely new set of applications which will become more requested in the years to come in industrialized nations. Aging population, increasing education levels, and a health-conscious culture will demand for robots that are

capable of assisting us in hard work, extend the autonomy of ill people, entertain and educate, and provide increasing levels of comfort.

In both cases, what matters most is the necessity to adapt to new and unpredictable situations, react quickly, display behavioral robustness, and operate in close interaction with human beings. Traditional robotic controllers designed for factory automation are not suitable for these types of applications. Evolutionary Robotics represent a powerful methodology to conceive robots that adapt to a changing environment with limited and sparse feedback. These applications demand relatively fast adaptation time and care-free operation. We think that the solutions suggested here of evolving adaptive controllers and use behavioral and internal fitness functions represent a step in this direction.

5 Acknowledgements

Dario Floreano acknowledges support from the Swiss National Science Foundation, grant nr. 620-58049. Joseba Urzelai is supported by grant nr. BF197.136-AK from the Basque government. Thanks to Patrick Saucy for his help on the cross-platform experiment.

Reference

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

Belew, R. K., & Mitchell, M. (Eds.). (1996). *Adaptive Individuals in Evolving Populations: Models and Algorithms*. Addison-Wesley, Redwood City, CA.

Chalmers, D. J. (1991). The evolution of learning: An experiment in genetic connectionism. In Touretzky, D. S., Elman, J. L., Sejnowski, T., & Hinton, G. E. (Eds.), *Proceedings of the 1990 Connectionist Models Summer School*. Morgan Kaufmann, San Mateo, CA.

Dorigo, M., & Colombetti, M. (1998). *Robot shaping: An experiment in behavior engineering.* MIT Press, Cambridge, MA.

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

Floreano, D., & Mondada, F. (1998). Evolutionary Neurocontrollers for Autonomous Mobile Robots. *Neural Networks*, *11*, 1461–1478.

Floreano, D., Nolfi, S., & Mondada, F. (2000). Co-evolution and ontogenetic change in competing robots. *Robotics and Autonomous Systems*. Forthcoming.

Floreano, D., & Urzelai, J. (1999). Evolution of Neural Controllers with Adaptive Synapses and Compact Genetic Encoding. In Floreano, D., Nicoud, J., & Mondada, F. (Eds.), *Advances in Artificial Life*. Springer Verlag, Berlin.

Gruau, F. (1994). Automatic definition of modular neural networks. *Adaptive Behavior*, *3*, 151–183.

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

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

Jakobi, N. (1997). Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In Husbands, P., & Harvey, I. (Eds.), *Proceedings of the 4th European Conference on Artificial Life* Cambridge, MA. MIT Press.

Kay, J., Floreano, D., & Phillips, W. A. (1997). Contextually Guided Unsupervised Learning Using Local Multivariate Binary Processors. *Neural Networks*, *11*, 117–140.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, *4*, 461–476.

Körding, K. and König, P. (2000). Learning with two sites of synaptic integration. *Network: Computation in Neural Systems*, *11*, 25–39.

Langton, C. G. (1990). Artificial life. In Langton, C. (Ed.), *Artificial Life*. Addison-Wesley: series of the Santa Fe Institute Studies in the Sciences of Complexities, Redwood City, CA.

Lewis, M. A., Fagg, A. H., & Solidum, A. (1996). Genetic programming approach to the construction of a neural network for a walking robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 2618–2623. IEEE Press.

Lund, H. H., & Miglino, O. (1998). Evolving and Breeding Robots. In Husbands, P., & Meyer, J. (Eds.), *Proceedings of the First European Workshop on Evolutionary Robotics*. Springer Verlag.

Mayley, G. (1996). Landscapes, Learning Costs and Genetic Assimilation. *Evolutionary Computation*, *4(3)*, 213–234.

Menczer, F., & Belew, R. K. (1993). Latent energy environments. In Belew, R. K., & Mitchell, S. (Eds.), *Plastic Individuals in Evolving Populations*. Addison Wesley, Redwood City, CA.

Miglino, O., Lund, H. H., & Nolfi, S. (1996). Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, *2*, 417–434.

Nolfi, S., & Floreano, D. (1999). Learning and evolution. *Autonomous Robots*, *7(1)*, 89–113.

Nolfi, S., & Floreano, D. (2000). *Evolutionary Robotics: Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA. Forthcoming.

Nolfi, S., Miglino, O., & Parisi, D. (1994). Phenotipic Plasticity in Evolving Neural Networks. In Nicoud, J.-D., & Gaussier, P. (Eds.), *Proceedings of the conference From Perception to Action*. IEEE Computer Press, Los Alamitos, CA.

Nolfi, S., & Parisi, D. (1995). Genotypes for neural networks. In Arbib, M. A. (Ed.), *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge, MA.

Rechenberg, I. (1973). *Evolutionstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Friedrich Fromann Verlag, Stuttgart.

Ronald, E., Sipper, M., & Capcarrère, M. S. (1999). Design, Observation, Surprise! A Test of Emergence. *Artificial Life*, *5(3)*, 225–240.

Thompson, A. (1998). On the automatic design of robust electronics through artificial evolution. In M. Sipper, D. M., & Prez-Uribe, A. (Eds.), *Proceedings of the 2nd International Conference on Evolvable Systems: From biology to hardware (ICES98)*, pp. 13–24. Springer-Verlag, Berlin.

Torgerson, W. (1958). *Theory and methods of scaling*. Wiley, New York.

Urzelai, J., & Floreano, D. (2000a). Evolution of Adaptive Synapses: Robots with Fast Adaptive Behavior in New Environments. Tech. rep. Technical Report, submitted for publication, LAMI-DI, Swiss Federal Institute of Technology in Lausanne.

Urzelai, J., & Floreano, D. (2000b). Evolutionary Robotics: Coping with Environmental Change. In Koza, J. e. a. (Ed.), *Second International Conference on Genetic and Evolutionary Computation* San Mateo, CA. Morgan Kaufmann.

Urzelai, J., & Floreano, D. (2000c). Evolutionary robots with fast adaptive behavior in new environments. In Fogarty, T. C., Miller, J., Thompson, A., & Thomson, P. (Eds.), *Third International Conference on Evolvable Systems: From Biology to Hardware (ICES2000)* Berlin. Springer-Verlag.

Widrow, B., & Hoff, M. E. (1999). Adaptive switching circuits. In *Proceedings of the 1960 IRE WESCON Convention*, Vol. IV New York. IRE.

Willshaw, D., & Dayan, P. (1990). Optimal plasticity from matrix memories: What goes up must come down. *Neural Computation*, *2*, 85–93.

Yamamoto, Y., Sasaki, T., & Tokoro, M. (1999). Adaptability of Darwinian and Lamarckian Populations toward an Unknown New World. In Floreano, D., Nicoud, J., & Mondada, F. (Eds.), *Advances in Artificial Life*. Springer Verlag, Berlin.

Yao, X. (1993). A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, *4*, 203–222.