Evolving and Breeding Robots

Henrik Hautop Lund **Orazio Miglino**

The Danish National Centre for IT Research – CIT
University of Aarhus, Ny Munkegade, bldg. 540, 8000 Aarhus C., Denmark
hhl@daimi.aau.dk
http://www.daimi.aau.dk/~hhl/

Faculty of Psychology
II University of Naples, Viale Giovanni Paolo I, 81055 S. Maria C. V. (CE), Italy
orazio@caio.irmkant.rm.cnr.it
http://gracco.irmkant.rm.cnr.it/orazio/

Abstract Our experiences with a range of evolutionary robotic experiments have resulted in major changes to our set-up of artificial life experiments and our interpretation of observed phenomena. Initially, we investigated simulation-reality relationships in order to transfer our artificial life simulation work with evolution of neural network agents to real robots. This is a difficult task, but can, in a lot of cases, be solved with a carefully built simulator. By being able to evolve control mechanisms for physical robots, we were able to study biological hypotheses about animal behaviours by using exactly the same experimental set-ups as were used in the animal behavioural experiments. Evolutionary robotic experiments with rats open field box experiments and chick detours show how evolutionary robotics can be a powerful biological tool, and they also suggest that incremental learning might be fruitful for achieving complex robot behaviour in an evolutionary context. However, it is not enough to evolve controllers alone, and we argue that robot body plans and controllers should co-evolve, which leads to an alternative form of evolvable hardware. By combining all these experiences, we reach *breeding robotics*. Here, children can, as breeders, evolve e.g. LEGO robots through an interactive genetic algorithm in order to achieve desired behaviours, and then download the evolved behaviours to the physical (LEGO) robots.

1 Introduction

In the field of evolutionary robotics, many researchers have tried to transfer a corpus of knowledge from previous experiences in evolving simulated artificial agents (e.g. controlled by neural networks [18, 24]) to model psychobiological phenomena. However, there are some difficulties that one has to consider. The first problem that researchers in this field had to face was how to avoid the extremely time consuming cost of genetic algorithms directly applied to real robots. Consider the following example, taken from some of our earlier experiments. In order to test a single mobile robot, it is allowed to run for 500 actions in an environment. In order to account for initial starting position biases, the mobile robot is run for 3 epochs with different starting positions. Since each action takes 100 m sec. to perform, the testing of the single robot's 1500 actions will take 150 sec. The evolution process of a population of 100 control systems for 100 generations will therefore take a minimum of 1,500,000 sec. or approximately 17 days (not accounting for the time spent on reproduction and movement of the robot to new initial positions). To report scientifically valid results, we would like to do the single test 5–10 times with different initial random seeds. This means that the robot has to run for half a year for us to obtain valid

results of just one experiment! Such a time consumption makes the approach infeasible, so researchers have tried to reduce it. We [16, 22] proposed to make part of evolutionary process in a simulated environment and a part in the physical environment. This approach required to: (a) specify a methodology for building an efficient simulator of the physical characteristics of robot and environment; (b) develop genotypes and techniques that produce plastic phenotypes that absorb the change of environments (e.g. from simulated to real environment). In section 2, we report our experiences with these problems.
However, after having built a robust setting to produce evolutionary robotic experiments, we had to confront the efficiency of the techniques to develop more complex behaviours, processes and architectures. In particular, section 3 reports some of our experiences (a) when simulating the behaviour of rats in a classical experimental setting, (b) simulating chicken detour behaviour, (c) evolving the hardware body structure. Thanks to a LEGO research and development contract, we recently explored the possibility of using evolutionary robotics in order to build an adaptive artificial pet as a toy for children. This experience (described briefly in section 4) represents the ultimate edge of our common work and convinces us that evolutionary robotics is rapidly becoming a sort of "Breeding Robotics". Our attempts to produce complex and adaptive behaviours made us orient the evolutionary process so that the pre-designing of environments, neural architectures, and fitness functions prompted the emergence of more sophisticated behavioural strategies. In substance, we coupled the power of an evolutionary process with the informality of a "breeder of robots".

2 Evolutionary Robotics Set-up

2.1 Evolving On-line or in Simulators?

As reported above, the on-line approach to evolutionary robotics means that the robot has to run for at least half a year for us to obtain valid results of just one experiment. Such a time consumption makes the approach infeasible, so researchers have tried to reduce it. One way to do so is to use a shorter time slice than 100 m sec. per action. This requires that the control system is not too complex, so that the appropriate motor responses can be calculated within this time slice. Another way is to test the robot for fewer actions — for example, testing the robot for only 200 actions for only 1 epoch will reduce the time consumption of a single evolutionary process in the example given above from 17 days to less than 2.5 days. This is the approach taken by researchers who do the evolution process entirely on the real robots (e.g. [4]). The approach has the disadvantage of not allowing many testings of each single control system (few actions and only 1 epoch), so the performance of a single control system will be heavily biased by the initial starting position of the robot.
Because of the problems with on-line evolution mentioned above, we have advocated a scheme to build simulators for robots, and then evolve the control systems in simulation before transferring the evolved control systems to the real robots in the real environments. Other researchers [10] have suggested using a mathematical description of motor and sensor responses of the specific robot. Their simulator is based on a set of equations, that should model the real world physics, which are used by the simulator to calculate specific values for the IR sensors and/or ambient light sensors and wheel speeds for each action of a specific robot. It is assumed that the responses of all sensors of a robot have the same characteristics, which is unfortunately not so! The individual sensors vary quite a lot from each other in identical conditions, so an accurate simulator must take the idiosyncrasies of each sensor into account. Despite the inaccuracy of a mathematical model, it can be used

effectively for experiments where high accuracy of transition of control systems from simulation to reality plays a minor role in the first instance. We have used such a mathematical model, where motor responses are decided by differential equations, in our studies of the evolution of robot body plans. In that study, the immediate goal is not to be able to transfer each single control system and robot body plan to a real robot, but in the end simply that of choosing the best controller plus body plan to be transferred to a robot built according to the evolved parameters.
Another methodology to build the simulator for a robot and its environment, is to build look-up tables of the robot's sensor and motor responses environments, as we proposed. By sampling the real world through the sensors and the actuators of the robot itself, one can build a quite accurate model of a particular robot-environment dynamics, and by using look-up tables constructed by this sampling instead of mathematical models of the sensor and motor responses, one obtains both very high performance when transferring the evolved control systems from simulation to the real robot, and a huge reduction in time consumption. The on-line approach takes 625 times as long, and the mathematical model simulator approach takes 3 times as long as the run with our look-up table approach.
The disadvantage of using a simulator based on the robot's own samplings of sensor and motor responses is that these responses have to be recorded. This can be done almost automatically with some robots in some simple environments, but it still has to be solved how to do this in more complex environments, where the construction of the look-up tables might prove more time expensive. It is simple to measure responses from sensors such as bumpers and black/white detectors, while it is more tricky for sensors such as infra-red sensors and cameras. Since each sensor is not placed in a pre-defined position on e.g. a LEGO robot, we have to run sampling procedures for each single sensor consecutively. When the positions of the sensors on the robot are pre-defined, e.g. on the Khepera robot and most other robots, the sampling can be done in parallel by taking samples from all sensors at the same time, but the sampling procedure has to be sequential when this is not the case. After taking samples from specific angles and distances with each of the robot's sensors, we can combine the look-up table with a kind of mathematical approach in which a mathematical description is used to calculate the entry in the look-up table for a specific position of the robot and a specific placement of the sensors on the robot. Further, to allow for non-symmetrical objects in an environment, samples must be taken with different angles of the different object-types towards each sensor. Therefore, the sampling procedure can be time-consuming in such cases.
The advantage of using this approach to build the simulator is that the sensor and motor responses are recorded by the robot itself. The look-up tables represent how the robot itself senses and moves in the environment, and not how we, as external observers, believe that the robot interacts with the real world. There is no need for symbolic or mathematical descriptions of objects or robot responses in this methodology. Therefore, the simulator becomes a very precise model of the real world situation, and the evolved control systems can be transferred successfully from the simulator to the real robot that interacts in the real world.

2.2 Evolving Neurocontrollers

In our first evolutionary robotics experiments [16, 22, 23], we were interested in investigating how to construct neural network control systems and how to develop these. We found that the most simple solution often turned out to be the best, namely in order to construct a neural network control system for a robot, one can simply use the robot's sensory activation as input stimuli for the neural network, and allow the output activation of the neural network to control the motors of the robot. To generate a simple feed-forward neural network control system for the

Khepera robot, one can connect the sensors with the motors, eventually with an intermediate layer of hidden units.
To simulate the evolution process that should develop the neural network control system for the Khepera robot, here, our version of the evolutionary robotics technique used a genetic algorithm (in other experiments, we used genetic programming). In our first experiments, we looked at the task of having the Khepera robot to perform an obstacle avoidance behaviour by moving forward as fast as possible, moving in as straight a line as possible and keeping as far away from objects as possible – a task that has now become a standard task in the field. In order to evaluate individual performance we used equation 1

$$F = \sum_{i=1}^{500} V_i(1 - DV_i^2)(1 - I_i) \quad (1)$$

where V_i is the average rotation speed of the two wheels, DV_i is the algebraic difference between signed speed values of the wheels, and I_i is the activation values of the proximity sensor with the highest activity at time i.
Since we wanted to address the problem of developing controllers within a reasonable time, we found it infeasible to run the whole evolution process on-line, and adopted a simulated/physical approach, where most of the evolution process ran in simulation.
For the obstacle avoidance task, we constructed the real, physical environment as a 60*35 cm rectangular arena surrounded by walls with 3 round obstacles of 5.5 cm placed in the centre. Walls and obstacles were covered with white paper. In order to construct a model of this environment, the Khepera robot empirically sampled the different classes of objects in the environment (wall and obstacles) through its own real sensors. The Khepera robot turned 360 degrees at different distances with respect to a wall and to an obstacle, while, in the meantime, recording the activation level of the sensors. The resulting matrices were then used by the simulator to set the activation levels of the simulated robot depending on its current position in the simulated environment. In the same way, to model the robot's motors, the effect of the different motors settings in the real world was sampled. For all possible states of the motors, it was modelled how the robot moved and turned. The obtained measures were used by the simulator to set the activation level of the neural network input units, and to compute the displacement of the robot in the simulated environment.
The procedure of letting the robot itself construct the model to be used by the simulator has several advantages: it is simple and it accounts for the idiosyncrasies of each individual sensor. It allows one to build a model of an individual robot taking into account the specificities of that robot that makes it different from other apparently identical exemplars. It also accounts for the idiosyncrasies of the environment by empirically modelling the environment itself, instead of building a mathematical model of it. We may however encounter problems such as objects of the same types may be perceived differently because of variation in the ambient light or because of slight differences in the objects themselves, actuators may produce unpredictable, uncertain effects, and the ground may present irregularities. Consequently, one would expect a decrease in performance when transferring a robot control system developed in a simulator to the real robot in the real world. Indeed, under "normal" conditions, a decrease in performance takes place when the control systems evolved in the simulator are transferred to the real robots in the real environment [22]. This is because of difficulties in constructing a simulator that is accurate enough to capture all the features of the real world that is important to the robot. To avoid this problem, noise can be applied in the simulated environment both to the simulated robots perception of objects and to the movement

of the simulated robot. The experiments with noise added to the simulator show a perfect mapping in performance from the simulated to the real environment (Fig. 2(a)). If, for instance, the evolutionary development of the neural network control system for the Khepera robot takes place for 200 generations in the simulator, and the neural network control system is transferred to the real Khepera robot in the real environment, then there will be no decrease in fitness score (performance) when the control system is transferred. Further, if the evolution continues for 20 generations in the real Khepera robot in the real environment, then there will be a slight increment in fitness score.
A two factor ANOVA (Fig. 1) applied to performances obtained by individuals at the transferring moment in the simulated and the real environment shows: (a) a statistical significance of transferring the control systems from the simulator with no noise to the real environment, and (b) no statistical significant effect of transferring the control systems from the simulator with noise to the real environment. This result supports the claim that the decrease in performance can be avoided by adding noise to the simulator. In metaphoric terms, adding noise permits the emergence of artificial "genotypes" that adapt to different environments.
It is remarkable, that at the level of behaviours, the simulated and the real robots

	Degrees of freedom	Ratio of Fisher	Interval of confidence	Remarks
No noise	1/998	164.14	0.001	Significant
Conservative noise	1/998	0.74	0.001	Not significant

Figure1. Results of two factors analysis of variance between performance of individuals at generation 200 in the simulated and in the real environment.

perform identically (Fig. 2(b)). The trajectories of the paths for the simulated and the real robots match almost perfectly. In this way, the evolutionary robotic technique suggests improvement of solutions to some old, well-known problems of robotics: how to construct an accurate model of the real environment, and how to develop control systems without explicitly designing submodules of the control system.

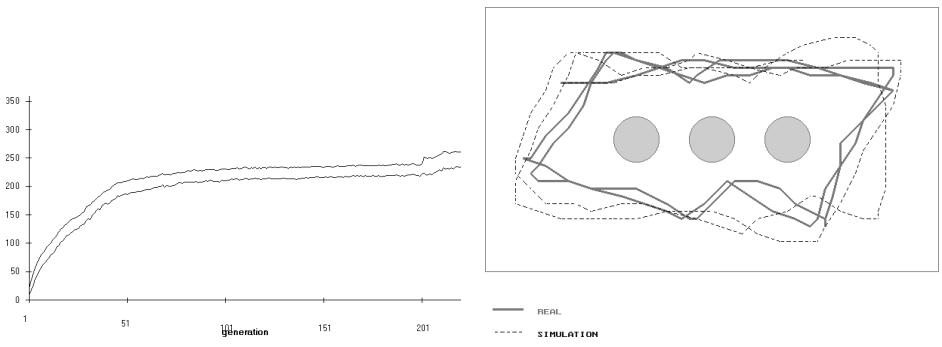

Figure2. (a) Peak and Average performance as an average result of 5 simulations. The first 200 generations are evolved in the simulated environment, the last 20 generations in the real environment. (b) Trajectory of the best individuals of generation 200 of a simulation in the simulated (dashed line) and real (full line) environment. © Miglino, Lund and Nolfi, 1995.

3 Complex Behaviours

Following our initial evolutionary robotics experiments that showed that it was indeed possible to evolve neural network controllers in simulation and then transfer them to the real robot in the real environment with no loss in performance, we naturally wanted to try to evolve more complex robot behaviours. First, we wanted to show how evolutionary robotics can be used as a tool in behavioural biology in order to verify/falsify hypotheses about animal control mechanisms. The examples that we used were the open field box experiments with rats and detour experiments with chicks.

3.1 Using Evolutionary Robotics to Test Conclusions of Experiments with Rats

During the last decade, researchers have used open field box experiments with rats in order to scientifically show that rats form cognitive maps to be used behaviourally in navigation tasks [1, 5, 20]. In open field box experiments, rats that have been shown the location of hidden food in a rectangular box are able to navigate towards and dig at that location (or at the rotational equivalent location) in a second identical box. Results like this lead researchers to state that rats use a cognitive map of the environmental shape. We studied a mobile robot in a similar experimental settings. Our "artificial organism" (i.e. robot) had no capability of constructing internal maps of the surrounding environment's geometrical shape. The robot, despite its sensory-motor limitations, performed the open field box experiments as well as rats. Our results show that the open field box experiments are not sufficient to conclude a construction of cognitive maps by the rats. Other task solutions based on no explicit knowledge of the environment are theoretically possible. In fact the geometrical properties of the environment can be assimilated in the sensory-motor schema of the robot behaviour without any explicit representation. In general, our work, in contrast with traditional cognitive models, shows how environmental knowledge can be reached withthout any form of direct representation (cognitive maps).
More precisely, what we did in these experiments, was to accurately replicate the animal behavioural experiments. A rectangular box was made out of wood and covered with white paper. As in the Gallistel experiments [5, 20], it measured 120*60 cm and was divided into 100 12*6 cm rectangles with lines on the floor (nine lines parallel to the box's short walls and nine lines parallel to the box's long walls). Likewise, the target area was a circular zone with a radius of 15 cm. A simple perceptron control system was constructed for the Khepera robot. By connecting the input of the 8 sensors to the 2 motors with traditional neural network connection weights, we obtain a simple perceptron (with linear output units). This simple control system does not have the capability of building an internal map of the surrounding environment, but it can simply make some sensory-motor responses. Again, we used a genetic algorithm to develop the controllers, as described above. The best performing neural network of the last generation (generation 30) was considered the final control system for the robot and, as in Gallistel's experiments, it was tested from 8 different starting positions: either at the middle of a wall, facing the box's centre or at the centre, facing the middle of a wall. The learning/testing process was repeated for each possible position of the target among the 80 intersections of the lines dividing the floor when excluding the centre intersection (point 5,5).
Even with as simple a control system as the perceptron for the Khepera, the robot is able to navigate to the target in the rectangular box. The table in figure 3 shows that as is the case with rats, the robot will however navigate to the rotational equivalent area as many times as to the correct target area. In 82% of the trials, the robots navigated to locations with the correct geometric relationship to the space defined

by the shape of the box, but half of the times it was at the rotational equivalent of the target area. The number of successes are comparable to the ones obtained with rats, and the robots perform less misses.

In the experiment with rats, the authors reported only the results on behavioural

	% Correct	% Rotational Errors	% Misses
Rats	35	31	33
Robots	41	41	18

Figure3. Percent of navigation to Correct Area, Rotationally Equivalent Area, and Incorrect Locations. Data on rats taken from [20].

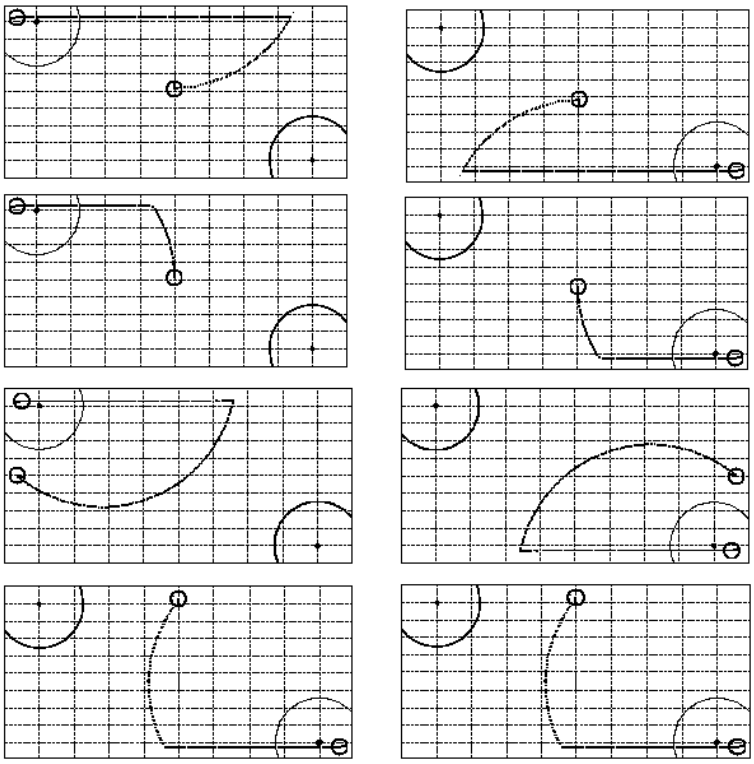

Figure4. The trajectory of the evolved Khepera robot in the open field box when started in the 8 starting positions. © Miglino and Lund , 1997.

indexes (percentage of correct loaclization, rotational error, and misses). Perhaps, a careful analysis of behavioural trajectories could give more information about cognitive processes involved in the task solution. Figure 4 presents typical Khepera trajectories for a target area located in a corner. In general, for all starting positions, the robot touches a long side of the box. If we did not know the strong sensorial limitations of Khepera we could hypothesise that Khepera discriminates long vs.

short sides. But, if we analyse figure 4 carefully, we can describe this strategy by a set of rules : 1) initially, when no sensors are active, Khepera produces a curvilinear trajectory on its left side; 2) when the robot touches the wall (the right-side sensors are active), then it turns left; and 3) Khepera goes on following the wall until it reaches a corner. Although the shape of the box (its geometrical characteristics) cannot be seen and represented by Khepera, it is assimilated in the behavioural sequences of the robot. For example, in order to touch a long side from every starting position, the radius of the initially curvilinear trajectory must be correlated with the size of the box.
These evolutionary robotics experiments contradict hypotheses based on animal behavioural experiments alone, and therefore the robotics experiments can be a powerful tool in biology. It suggests that animal psychologists should not only report behavioural indexes but also analyse, with attention, the behavioural trajectories of the animals or to provide neurophysiological evidence that account for hypothesised activation of cerebral modules in such experiments.

3.2 Towards Complex Behaviours: Incremental Learning

Another example of the use of evolutionary robotics to test animal behavioural hypotheses is found in detour behaviour experiments. We will here look at this example, but put emphasis on how the robot detour experiments might help us achieving more complex behaviours with the evolutionary technique.
Animal psychology literature has intensively investigated detour behaviour in several species of mammals [12, 32]. In substance, a detour task consists in reaching a target following trajectories that, for specific environmental constraints (obstacles, different light conditions, etc.), deviate from an optimal approaching path.
One of the most used settings to observe detour behaviour is represented by a situation where the target is placed behind a U-shaped barrier (see figure 5b). In this case, in order to reach the target, an organism must make a detour around the obstacle and therefore in some input-output cycles it loses sight of the target. An organism which is able to make a detour around an obstacle is said to posses a representation of the spatial environment [28]. In fact, if the organism is able to negotiate the obstacle and reach the target, this implies that the organism is in possession of a spatial representation of the environment that tells the organism where the target is even if the target is currently not visually accessible.
Two-day-old chicks perform quite well on a detour task with a U-shaped barrier [28, 34]. Unfortunately, the complexity of natural experimental settings obliges researchers to deduce neural mechanisms and cognitive processes underlying detour abilities by just monitoring behavioural indexes (time spent in a particular area, time spent to reach a target, etc.). In fact, the question about how chicks solve the detour task is still open: "... Chicks could have used a motor algorithm with instruction such as if you turned right (left) before the goal disappeared, then turn right (left) to find it again. Dead reckoning could be a possibility: chicks may be able to continuously update their position with respect to the goal in a represented space moment by moment..." [28] pp. 4. Given this state of the art in animal spatial behaviour, we think that evolutionary robotics could be useful to give insights and computational models. In fact, by designing robotics experiments that reproduce exactly the animal experimental setting it is possible study "a posteriori" the computational structures that emerged through the learning/evolutionary process and confront them with the strategies of real organisms.
According to that perspective, we evolved different neural controllers (i.e. different neural architectures: perceptron, with hidden layer, with Elman memory units, etc.) for the Khepera robot equipped with a linear camera (for details see [21]). The environment where Khepera acted was a rectangular walled arena of 55x40 cm. It

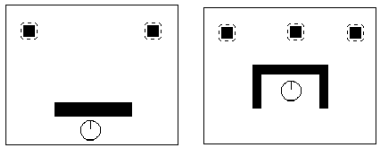

Figure5. The environments used in detour experiments: (a) in the first 200 generations; (b) after 200 generations. The dotted areas identify alternative target positions.

contained a low obstacle that had a U-shaped obstacle and a cylindrical target (see figure 5b). The proximity sensors of the robot can sense the obstacle or the target or the peripheral wall provided the robot is sufficiently close to them. The camera can see the target (but not the obstacle or the wall) at any distance (provided the target falls within its restricted visual field) even when the target is on the other side of the low obstacle. At the beginning of each trial, Khepera was positioned in the environment behind the obstacle facing North. The target was positioned beyond the obstacle. The fitness function was based on two components. The first component was how much time Khepera spent in each trial near the target, where nearness to the target means to be located 12 cm or less from the target. The second component is calculated according to formula (1) mentioned in section 2.2, however where only three IR sensors where used. In other words, the second component of fitness rewards the individual, in each step, if the individual moves fast and straight (without turning) and it punishes the individual for being too close to the obstacle or wall (i.e., when the infrared sensors are activated; notice that the second fitness component is ignored when the robot is near the target).
The situation described above reflects the characteristics of the detour task used in animal experiments. In fact, the only solution for the robot is to turn right or left and make a detour around the wall in order to reach the target. But this creates a serious problem for our robot. By turning to its right or left, the robot is turning away from the target and therefore the target is likely to leave the robot's visual field. The problem is particularly serious if the obstacle has the shape shown in figure 5. In order to negotiate the obstacle, the robot may be forced to go in the opposite direction with respect to the target and therefore to loose sight of the target.
After many experiments, we reached a corpus of data that helped analysing the power and limitations of different emerged structures (i.e. neural architectures). Feed-forward networks were able to make detour behaviour, but only recurrent networks (with Elman memory units) produced optimal solutions (for details see [21]). Because of space limitations, here we only want to stress how we "obliged" Khepera to learn the task. In a first series of experiments, we put Khepera directly in an environment containing a U-shaped barrier (such as in figure 5b). In this condition and with different neural architectures, Khepera was not able to "learn" the task. Then, we adopted a strategy to make a form of incremental learning.
For the first 200 generations, Khepera "lived" in an environment that contained a rectangular obstacle (a bar of 20 cm of length and 3 cm of height) and a target (a cylinder of 2 cm of diameter, see figure 5a). Each trial lasted a total of 240 input/output cycles divided into two epochs of 120 cycles each. At the beginning of each epoch, the Khepera was positioned in the environment behind the obstacle facing North. The target was positioned beyond the obstacle once on the left side and once on the right side of the upper portion of the environment in such a way that the target fell outside the organism's restricted visual field and therefore it could not be perceived by the robot. Khepera had 120 cycles available to make a

detour around the obstacle and reach the target.
After 200 generations, the environment changed. The obstacle now had an inverted U-shape and was moved slightly toward North (see figure 5b). The robot was placed behind the obstacle facing North at the beginning of each epoch as in the preceding environment. Each trial lasted 6 epochs (of 120 cycles each). In 3 of the 6 epochs there was no obstacle. In the other 3 epochs, the target was located once on the left side, once in the right side, and once in the middle of the upper portion of the environment. This environment management (notice that we only changed the environmental characteristics and not the fitness formula) drove Khepera to perform the detour task in a similar way to the chicks.
In short, it seems possible to reach complex behaviours with a careful design of the evolutionary environmental conditions. However, this practise needs more theoretical investigations. A first step in that direction is represented in [2].

3.3 Evolving Robot Morphology

Recently, there has been an increasing interest in evolvable hardware (EHW), but the concept does not seem yet to be well-defined. There is a lot of discussion within the EHW community on whether to evolve at gate-level or function-level, use genetic programming or cellular programming, do online evolution or off-line evolution. In our view, the best definition of EHW that includes all of the above, is the one by Yao and Higuchi [36], who defines EHW to be hardware that can change its architecture and behaviour dynamically and autonomously with its environment, and they advocate that "EHW should be regarded as an evolutionary approach to behaviour design rather than hardware design" [36].
A great number of experiments have been performed in EHW, in which the emphasis has been to show the validity of evolving electronic circuits [6–9, 31]. A quite different approach to EHW, namely an evolvable electro-biochemical system, has been suggested by Kitano [11]. There has been some initial approaches using EHW in robotics tasks, most notably by Thompson [33], and if one looks at the definitions of autonomous robots and of EHW, they appear indistinguishable: hardware that can change its architecture and behaviour dynamically and autonomously with its environment.
The traditional applications of EHW in robotics only evolved the control circuit architecture of the robots. However, the circuit architecture is only a part of the hardware system, and ideally we would like to evolve the whole system. The hardware of a robot consists of both the circuit, on which the control system is implemented, and the sensors, motors, and physical structure of the robot.
We [15] have previously argued that *True EHW* should evolve the whole hardware system, since the evolution and performance of the electronic hardware is largely dependent on the other parts of the hardware that constitute the system. The latter part is what we called the *robot body plan*. A robot body plan is a specification of the body parameters. For a mobile robot, it might be types, number and position of sensors, body size, wheel radius, wheel base, and motor time constant. With one specific motor time constant, the ideal control circuit should evolve to a different control than with another motor time constant; different sensors demand different control mechanisms; and so forth. Further, the robot body plan should adapt to the task that we want the evolved robot to solve. An obstacle avoidance behaviour might be obtained with a small body size, while a large body size might be advantageous in a box-pushing experiment; a small wheel base might be desirable for a fast-turning robot, while a large wheel base is preferable when we want to evolve a robot that turns slowly; and so forth. Hence, the performance of an evolved hardware circuit is decided by the other hardware parameters. When these parameters are fixed, the circuit is evolved to adapt to those fixed parameters that, however,

might be inappropriate for the given task. Therefore, in *true EHW*, all hardware parameters should co-evolve.
The definition of *true EHW* is in accordance with the above mentioned definitions of EHW and autonomous robots that classify such systems as being able to change their architectures and behaviours dynamically and autonomously with their environment. Indeed, the design of a complete autonomous system must include self-organisation, and self-organisation means adaptation of *both* morphology and control architecture.
Together with Lee and Hallam, we [13, 15] therefore investigated the possibility of co-evolution between robot controllers and robot body plans in simulation, and we are currently building a LEGO robot system that allows evolution of robot body plans for real robots. The experiments showed that controllers and robot body plans are tightly coupled and that the robot body plan is determining for the performance of a controller.
We also went a step further and constructed specific hardware pieces, that would allow us to study the evolution of the morphology on a physical robot. We built a set of ears with hardware programmable pre-amplifiers, delays and mixers for the Khepera robot [19]. The hardware allows us to approximate the auditory morphology of various crickets by adjusting the programmable delays and the summing gains. With the reconfigurable hardware, we can investigate the relationship between the auditory morphology, the con-specific song, and the internal control system that generates the phonotaxis behaviour shown by the female cricket in response to the call of a mate. One possible investigation is then to co-evolve controller and auditory morphology to give good phonotaxis to a specific song while giving good discrimination between different kinds of songs.

4 Towards Breeding Robots

In most evolutionary robotics studies, there is the underlying assumption that it is in some sense trivial to design the mathematical fitness functions that can be used by the evolutionary algorithm to guide the development of robot controllers for task achieving behaviours. Yet, in some cases it might be difficult to think of the right fitness function before doing empirical testing. Recent experiments (see above section 3.2 and [25, 27]) do indeed suggest that some kind of shaping is necessary in the process of evolving task achieving robot behaviours, and other important results regarding shaping is found in other robotic directions (see e.g. [3]). The difficulties in describing adequate fitness functions (based on extensive studies of the problem domain), are seen by us as one of the main reasons why practically no researchers in the evolutionary robotics field have yet achieved truly complex robot behaviours.
Thanks a LEGO research and development contract, we explored the possibility to allow children to develop robot controllers for their individual "needs" or "taste". Obviously, we could not use the traditional evolutionary robotics approach, since it would require the children to have knowledge about how to design mathematical fitness functions.
We also wanted an approach different from the traditional *LOGO programming of LEGO-robots [29, 30]. Between these two constraints (traditional programming vs. evolutionary robotics), we chose a third way. The general idea is an *Interactive Evolutionary Robotics* approach by which children can develop (or evolve) robot controllers in the simulator by choosing among different robot behaviours that are shown on the screen, and then, when they are satisfied with the simulated robot's behaviour, download the developed control system to the real LEGO robot and further play with it in the real environment.
The interactive evolutionary robotics approach is inspired by our previous work

using interactive genetic algorithms to evolve simulated robot controllers, facial expressions and artistic images (see e.g. [26, 35]). In this approach, there is no need of programming knowledge, since all the end-user has to do is to choose between the solutions suggested graphically on the screen. Hence, there is no description of a fitness function, but the selection in the genetic algorithm is performed by the user. Surprisingly, we observed that children, using our tool, have been able to produce most of the simple robot behaviours that have been developed by researchers in the evolutionary robotics field.
We started to build a couple of LEGO robots for testing purposes, one of which is shown in Figure 6 (for details see [17]). As mentioned above and extensively re-

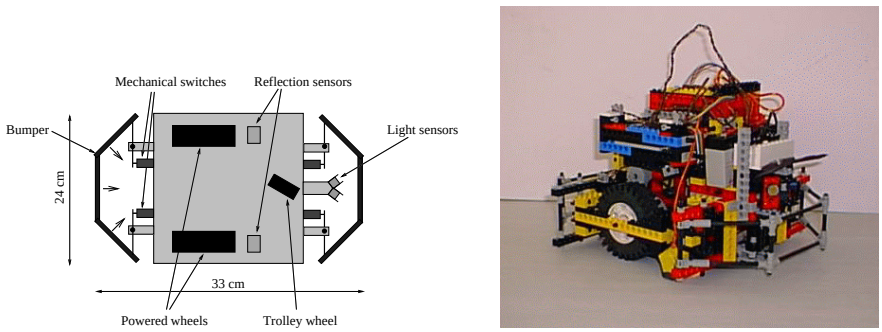

Figure6. Left: Schematic (upper) view of the jeep-like LEGO robot. The robot had three wheels: two powered wheels for locomotion and steering and one passive trolley wheel. The robot received sensory information from 4 mechanical switches mounted within two bumpers, 2 ambient light sensors at its front, and 2 reflection detectors under the chassis. Right: Photo of the LEGO robot. © Lund et al. 1997.

ported in [14, 16, 22], it is possible to build an accurate simulator that allows one to go from simulation to reality by basing the simulator on the robot's own samplings of sensor and motor responses. The LEGO simulator was built by this technique (described in section 2).
The architecture of the control system that calculates the motor output given a sensory input in the simulated and real LEGO robots was a feed-forward neural network that connected sensory input to motor output. The jeep-like LEGO robot had 8 input neurons (4 bumpers and 4 light sensors), and the output was coded in 6 output neurons. In order to obtain a controller that guides the LEGO robot to perform a desired task, one will have to choose the right values for the (in this case) 42 connection weights in the feed-forward neural network control system. In a sense, this corresponds to finding a specific point (or region) in a 42 dimensional space. This is, of course, impossible for children (and most adults). Therefore, we used an interactive genetic algorithm to develop these connection weights. This algorithm allowed children to develop the connection weights (i.e. the control system) by choosing among robot behaviours shown on the computer screen. The interactive genetic algorithm in the simulator works as follows.
The simulator allows different population sizes, but let us here look at a population of 9 robots. These 9 robots are placed in each their copy of the environment. The 9 similar environments are shown on the screen at the same time. Initially, the 9 robots in the population (generation 1), may have connections weights generated at random or loaded from a set of previously saved connection weights. The first of the 9 robots is put into the first environment. The robot will produce reactions based on the connection weights of the neural network control system and the sensory inputs.

These reactions (movements) of the robot are shown in the first environment on the screen. After this robot has run around for a given time, the second robot is put into the (similar) second environment, and the produced movements are shown on the screen. This is done for all 9 individual robots. Hence, at this moment, the child has seen the 9 robots moving around in each their environment one after another. Let us look at the example where the robots' neural network control systems are generated at random. Some of the robots will differ in their behaviour, since the connection weights in the neural networks differ. The child might like some of these behaviours and dislike others. Based on such an evaluation where the child show preferences, the preferred robots in the population can be chosen to reproduce. The child can, for instance, choose 3 different ones, or choose the same robot 3 times.
In the reproduction phase, the 3 selected robots will be copied 3 times each in order to produce the next generation (generation 2) of 9 robots. However, as the neural network control systems are copied, mutation will be applied to 10% of the connection weights chosen at random. If a weight is selected, it will have added a value chosen at random in the interval [-10,10]. Hence, the new generation of robots will be similar to their parents because they are copies, but they will also differ to some degree because of the mutation.
The new generation of 9 robots is shown on the screen, one robot after another, as was the case with the previous generation. The child can then again select the 3 preferred robots and they will reproduce through copying and mutation to make the next generation (generation 3). This loop can continue until the child is satisfied with the behaviour of the LEGO robot. It is important to note, that the way the LEGO robots change is based on the child selecting which ones to reproduce according to the child's individual preferences.
Another way to change a LEGO robot's behaviour is to develop a single LEGO robot's behaviour rather than develop on a whole population. If this option is chosen, an enlarged image with the environment and robot will appear on the screen (see Figure 7 (right)). The robot will start moving around in this environment, just as it did when the whole population was presented. Now, if the child dislikes a movement of the robot (e.g., the robot gets stuck when hitting into a wall), the child can press a "Bad" button. This will immediately change the underlying neural network control system of that robot. But, in this case, the change does not happen at random. The system keeps a record of the last input to the robot, and it is the connections weights going out from the input neurons that were activated that are changed (with random portions). In the case where the robot hit into the wall with the front left bumper, it will be the connection weights going out from the input unit for front left bumper that are changed. Hence, the reaction upon a bump with the front left bumper is changed. Each time the "Bad" button is pressed, there is also a slight chance of the bias connections being changed, resulting in a change in the default behaviour.
When the child is satisfied with the behaviour obtained in the teaching process, the child can return to the whole population. The trained robot keeps the trained network, and can then, eventually, be selected to reproduce or to download on the real LEGO robot. In this way, we allow a Lamarckian evolution that consists of characteristics being learnt during life-time being inherited over evolution. We by no means support such a theory for natural evolution, but as a means for fast development of LEGO robots, it has proved to be a very powerful tool.
In a sense, what this type of reinforcement learning does is to make a guided search in specific regions. In the case with the bump with the front left bumpers, the algorithm performs a search on the weights connecting that input unit to the output units, and the search is made only on these weights. It seems to quickly find a solution, because the dimension of the search space has been limited. This is not a traditional reinforcement learning algorithm, but for the simple tasks of the LEGO

robots we found that it works very well. We performed a series of experiments with

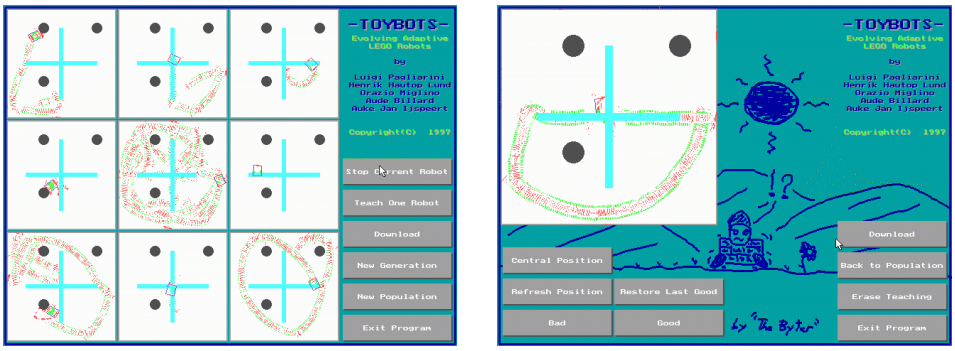

Figure7. Left: The population of 9 robots placed in each their (similar) environment. Here, the environment contains a couple of lines on the floor (the cross) and three round obstacles. In each environment, one can see the trajectory of the robot that has moved around in this environment. Right: The enlarged environment in which the child can train a single Toybot through reinforcement learning. The control system of the Toybot is changed every time the "Bad" button is pressed. © Lund et al. 1997.

the interactive evolutionary robotics approach described above, and had a lot of children playing with the LEGO robots. As expected, it turns out to be very simple to evolve exploratory behaviours, different kinds of obstacle avoidance, line & wall following, etc. and combinations between such simple behaviours. Hence, within minutes, children can develop a wide range of the robot behaviours that have traditionally been evolved in the field of evolutionary robotics. This is due to the use of the interactive evolutionary robotics approach combined with reinforcement learning. Indeed, with this approach children do not need to have any programming skills whatsoever! All that is demanded is that the child specify individual preferences in order to develop robots according to the individual taste.

5 Conclusions

In this paper, we have outlined part of our joint work in evolutionary robotics during the past years. We have moved from artificial life simulation work to real robot experiments and showed how this is possible with a carefully designed robot simulator. This allowed us to test animal behavioural hypotheses directly by evolving real, physical agents to perform in identical experimental set-ups. These experiments indicated that some kind of shaping would be necessary when evolving complex robot behaviours, and indeed we are now introducing *breeding robotics* that combines shaping and evolutionary robotics by using interactive genetic algorithms and reinforcement learning. Our future efforts will aim at showing how truly complex behaviours can emerge from breeding robots.

Acknowledgements.

Materials and facilities were provided by the LEGO Group, the Danish National Centre for IT Research – CIT, the National Research Council, Rome, Italy, and the University of Edinburgh. Thanks to co-authors of the referenced papers (Billard, Denaro, Hallam, Ijspeert, Lee, Nafasi, Nolfi, Pagliarini, Parisi, Tascini, Taylor, Vucic, Webb).

References

1. K. Cheng. A purely geometric module in the rat's spatial representation. *Cognition*, 23:149–178, 1986.
2. A Clark and C. Thornton. Trading Spaces: Computation, Representation, and Limits of Uniformed Learning. *Behavioral and Brain Sciences*, 20:57–90, 1997.
3. M. Dorigo and M. Colombetti. Robot Shaping: Developing Autonomous Agents through Learning. *Artificial Intelligence*, 71(2):321–370, 1994.
4. 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 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB94)*, Cambridge, MA, 1994. MIT Press, Bradford Books.
5. C. R. Gallistel. *The organization of learning.* MIT Press, Cambridge, MA, 1990.
6. H. Hemmi, J. Mizoguchi, and K. Shimohara. Development and evolution of hardware behaviours. In R. Brooks and P. Maes, editors, *Proceedings of ALIFE IV*, Cambridge, MA, 1994. MIT Press.
7. T. Higuchi, H. Iba, and B. Manderick. Evolvable Hardware with Genetic Learning. In H. Kitano, editor, *Massively Parallel Artificial Intelligence*, MA, 1994. MIT Press.
8. T. Higuchi, M. Murakawa, M. Iwata, I. Kajitani, W. Liu, and M. Salami. Evolvable Hardware at Function Level. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, NJ, 1997. IEEE Press.
9. T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya. Evolving hardware with genetic learning: A first step towards building a Darwin machine. 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 (SAB92)*, Cambridge, MA, 1992. MIT Press-Bradford Books.
10. N. Jakobi, P. Husbands, and I. Harvey. Noise and The Reality Gap: The Use of Simulation in Evolutionary Robotics. In *Proceedings of 3.rd European Conference on Artificial Life (ECAL'95)*. Springer-Verlag, 1995.
11. H. Kitano. Towards Evolvable Electro-Biochemical Systems. In C. Langton and T. Shimohara, editors, *Proceedings of Artificial Life V*, Cambridge, MA, 1997. MIT Press.
12. W. Koelher. *The Mentality of Apes.* Harcout Brace, New York, 1925.
13. W.-P. Lee, J. Hallam, and H. H. Lund. A Hybrid GP/GA Approach for Co-evolving Controllers and Robot Bodies to Achieve Fitness-Specified Tasks. In *Proceedings of IEEE Third International Conference on Evolutionary Computation*, NJ, 1996. IEEE Press.
14. H. H. Lund and J. Hallam. Sufficient Neurocontrollers can be Surprisingly Simple. Research Paper 824, Department of Artificial Intelligence, University of Edinburgh, 1996.
15. H. H. Lund, J. Hallam, and W.-P. Lee. Evolving Robot Morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, NJ, 1997. IEEE Press. Invited paper.
16. H. H. Lund and O. Miglino. From Simulated to Real Robots. In *Proceedings of IEEE Third International Conference on Evolutionary Computation*, NJ, 1996. IEEE Press.
17. H. H. Lund, O. Miglino, L. Pagliarini, A. Billard, and A. Ijspeert. Evolutionary Robotics — A Children's Game. In *Proceedings of IEEE Fifth International Conference on Evolutionary Computation*, NJ, 1998. IEEE Press.
18. H. H. Lund and D. Parisi. Pre-adaptation in populations of neural networks evolving in a changing environment. *Artificial Life*, 2(2):179–197, 1996.
19. H. H. Lund, B. Webb, and J. Hallam. A Robot Attracted to the Cricket Species *Gryllus bimaculatus.* In P. Husbands and I. Harvey, editors, *Proceedings of Fourth European Conference on Artificial Life*, pages 246–255, Cambridge, MA, 1997. MIT Press, Bradford Books.
20. J. Margules and C. R. Gallistel. Heading in the rat: Determination by environmental shape. *Animal Learning and Behavior*, 16:404–410, 1988.
21. O. Miglino, D. Denaro, G. Tascini, and D. Parisi. Detour Behavior in Evolving Robots. 1998. In these proceedings.

22. O. Miglino, H. H. Lund, and S. Nolfi. Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, 2(4):417–434, 1996.
23. O. Miglino, K. Nafasi, and C. Taylor. Selection for Wandering Behavior in a Small Robot. *Artificial Life*, 2(1), 1995.
24. O. Miglino, S. Nolfi, and D. Parisi. Discontinuity in evolution: how different levels of organization imply pre-adaptation. In R. K. Belew and M. Mitchell, editors, *Adaptive Individuals in Evolving Populations: Models and Algorithms*. Addison-Wesley, 1996.
25. S. Nolfi and D. Parisi. Evolving non-trivial behaviors on real robots: A garbage collecting robot. *Robotics and Autonomous Systems*, 1997. To appear.
26. L. Pagliarini, H. H. Lund, O. Miglino, and D. Parisi. Artificial Life: A New Way to Build Educational and Therapeutic Games. In C. Langton and K. Shimohara, editors, *Proceedings of Artificial Life V*, pages 152–156, Cambridge, MA, 1997. MIT Press.
27. S. Perkins and G. Hayes. Incremental acquisition of complex behaviour using structured evolution. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms '97*. Springer-Verlag, 1997. To appear.
28. L. Regolin, G. Vallortigara, and M. Zanforlin. Object and Spatial Representations in Detour Problems by Chicks. *Animal Behavior*, 48:1–5, 1994.
29. M. Resnick. MultiLogo: A Study of Children and Concurrent Programming. Master thesis, MIT, 1988.
30. M. Resnick. LEGO, Logo, and Life. In C. Langton, editor, *Artificial Life*, MA, 1989. Addison-Wesley.
31. M. Sipper, M. Goeke, D. Mange, A. Stauffer, E. Sanchez, and M. Tomassini. The Firefly Machine: Online Evolware. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, NJ, 1997. IEEE Press.
32. C. Thinus-Blanc. *Animal Spatial Cognition*. World Scientific, Singapore, 1996.
33. A. Thompson. Evolving electronic robot controllers that exploit hardware resources. In F. Moran, A. Moreno, J.J. Merelo, and P. Charon, editors, *Advances in Artificial Life: Proceedings of 3.rd European Conference on Artificial Life*, Heidelberg, 1995. Springer-Verlag.
34. G. Vallortigara and M. Zanforlin. Position Learning in Chicks. *Behavioral Proceedings*, 12:23–32, 1986.
35. V. Vucic and H. H. Lund. Self-Evolving Arts — Organisms versus Fetishes. *Muhely - Hungarian Journal of Modern Art*, 104:69–79, 1997.
36. X. Yao and T. Higuchi. Promises and Challenges of Evolvable Hardware. In *Proceedings of First International Conference on Evolvable Systems: from Biology to Hardware*, Heidelberg, 1996. Springer-Verlag.