Evolutionary Robotics in Behavior Engineering and Artificial Life

DARIO FLOREANO
Laboratory of Microcomputing
Swiss Federal Institute of Technology
CH-1015 Lausanne
Dario.Floreano@epfl.ch

1 Introduction

Since the first edition of the ER conference in 1993, Evolutionary Robotics has become a well-established discipline with several groups worldwide continuously reporting new theoretical and practical advancements. Two different directions have emerged during these five years within the ER community. In one case the focus is rather on engineering applications of adaptive robotics. In the other case, the interest consists in investigating theories and models of biological and artificial life. An informative measure of the extent to which an experiment belongs to either approach is given by the amount of external control put into the selection criterion (fitness function) and architecture parameters. Whereas industrial applications are based on well-defined goals and constraints, biological organisms evolve without an externally-imposed selection criterion and even the notion of evolutionary progress is controversial.

In this chapter, I will present examples of these two methodologies in a descriptive and informal fashion (the interested reader is referred to the technical reports available in published elsewhere). In the first example, careful evolution is used to incrementally develop modular control architectures that achieve complex tasks in the real world through a *shaping* procedure. This method can be readily adopted for real-world applications. In the second example, two robots in competition with each other (a predator and a prey) quickly coevolve a variety of behavioral strategies under selective pressure dictated by the competitor. It is shown that *unplugged co-evolution* develops highly competitive robots that do not necessarily maximize intuitive fitness functions. Far from representing a dichotomy, these approaches indicate the variety and richness of artificial evolution in robotics.

2 Robot training

When it comes to industrial applications of robotics, the most common approach still consists of constraining as much as possible the environment where the robot will operate; create a model of it and of the robot; and devise a program that will reliably and precisely guide the robot through a sequence of predefined steps. Programming robots using this methodology is very difficult (therefore, very costly) and the resulting program is very brittle and not generalizable to other robotic platforms or to a different environment. It is clear that a different approach is required if one wants to bring robots into everyday life (offices, houses, hospitals, roads, etc.), create more robust systems that do not require too many modifications to the layout of the environment where the robot will operate, and save money in terms of dedicated equipment and man/hours for developing a particular application.

Although in this manuscript I will concentrate mainly on control problems, it is important to stress that a new approach requires also a different way of building robots (e.g., [13]). In the traditional approach, robots are often closed systems (they cannot be easily expanded or modified) tailored for a specific task and operating condition, and rely on precise, costly, and fragile components. These are consequences of a Cartesian approach that attempts to fit a mathematical model and aspire to precision of measurement and of movement [18]. If robots are to enter everyday life and be capable of adapting to their surroundings, perhaps by trials and errors, they must be built out of cheaper components, be sturdy, modular, and recyclable (e.g., re-adaptable for a new use).

Let us consider the case of autonomous mobile robots. The appeal of these machines is that they are capable of operating in partially unknown and dynamic environments. By *partially unknown* I mean that the engineer who develops the robot and its control system is not aware of all the details of the environment that might affect the robot behavior. By *dynamic environment* I mean that some characteristics of the environment might change at some point while the robot is operative. Rodney Brooks clearly showed that a completely different methodology based on behavioral decomposition, rather than functional decomposition, can be used to develop and debug very quickly efficient control programs in close interaction with the environment [4, 5, 6]. Behaviors are simple modules that receive sensory input and can produce motor outputs. These modules can be combined together by simple local rules in order to achieve a desired complex behavior. The idea is that when something changes in the environment, it is always possible to add a new module or modify an existing one by observing the behavioral result.

Several researchers have proposed to add learning abilities to cope with the uncertainties of the environment (e.g., see [10, 17, 23, 24] for some recent work). Learning can be useful not only for making more autonomous the acquisition of behavioral skills, but also for maintaining a constant performance level of simple sensorimotor behaviors in dynamic environments. Within this framework, Dorigo and Colombetti [11] present an engineering methodology for designing adaptive autonomous robots, that is robots which can be trained (rather than programmed) to carry out useful tasks in close interaction with their operating environment in a reasonable amount of time and reliably. In the effort to make the training process autonomous, the reinforcement value is automatically computed on a step-by-step basis using only information available to the robot from its own sensors. The learning phase takes place incrementally and goes through several phases assisted by a human operator. Modularity is ubiquitous also in the methodology used by Dorigo and Colombetti: it is present in the phase of behavioral decomposition, in the choice of control architectures, and in the design of shaping policies. However, the problem with designing modular systems is similar to that faced by an orchestra director who attempts to obtain an harmonious symphony out of several musicians playing different instruments, that is how to coordinate all the different elements. Dorigo and Colombetti have a simple recipe for keeping their orchestra together—sum, combine, suppress, and sequence, but, as they acknowledge, more powerful methods are required for scaling up to complex behaviors. In the next section, I will show how evolution can play the role of this orchestra director.

3 Evolutionary Robot Training

We have combined a modular, behavior-based, incremental approach with learning techniques. In our methodology, learning takes place at two levels: at the level of the architecture and at the level of each individual module. At the architectural level, an evolutionary algorithm coordinates the activation of several behavioral modules; at the module level, reinforcement learning algorithms self-supervise the acquisition and reliability of behavioral skills using only information that is directly and immediately available to the robot and to that specific module. Architectural evolution is more concerned with the incorporation of new major skills which are roughly defined by a human operator. Module learning, instead, is concerned with fine adjustements of individual modules to relatively-minor environmental changes. However, since learning and evolution do interact [16], changes at one level can percolate in both directions through the architecture to the other level so that the overall performance remains stable and reliable. Finally, the architecture can be built and evolved incrementally by using a simple type of variable-length genetic representations.

The control architecture employed is composed of a set of fully interconnected modules. Each module (figure 1) has an input and an output. The input comes from the sensors of the robot. The output consists of two messages: an activation level that indicates whether the module wants to affect the robot current action and an output vector which consists of a motor command.

The internal structure is based on two components: an *activation network* and a *behavior generator*. The activation network decides the activation level of the module by combining current sensory information with the weighted sum of activation signals coming from other modules. The resulting activation level is sent out to other modules in the architecture. Connections among modules have excitatory or inhibitory values. At each time step, a Winner-Take-All process occurs among modules. The module winning this competition will access the motor resources and control the robot for a short time slice. In our experiments the activation network is implemented as a feedforward neural network. The behavior generator can be a pre-programmed behavior, a neural

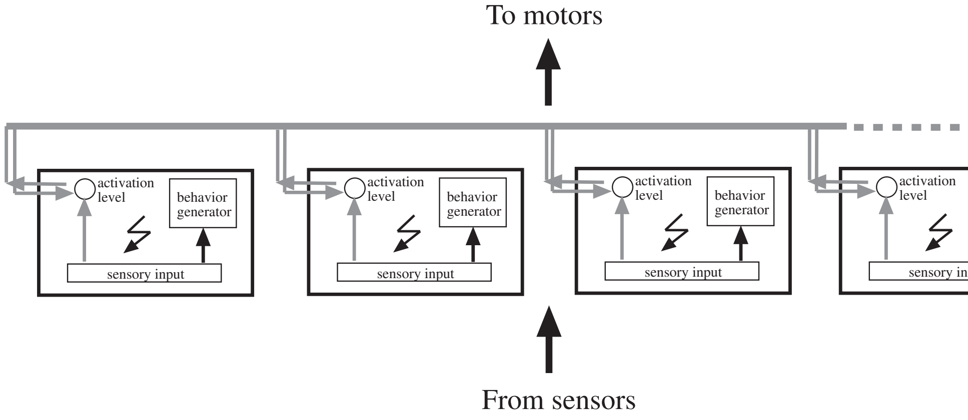

Figure 1: Each module is composed of an activation network (which decides the activation level of the module) and of a behavior generator (which implements the behavior). The module also includes a local learning algorithm that can act both on the activation network and on the behavior generator. The activation level is computed using weighted signals from the available sensors and from other modules. A Winner-Take-All procedure decides every 300 ms which module controls the robot. Light grey connections are evolved.

network, a classifier system [3], or any other structure suitable for generating motor commands and other behavioral decisions in response to sensory inputs.

A module can incorporate also a local learning algorithm and a *Reinforcement Program*. The reinforcement program is a function that transforms sensorimotor inputs into negative and positive reinforcement signals. In our architecture, only immediate reinforcement signals are used. The learning algorithm can adapt both the parameters of the activation network and of the behavior generator using a reinforcement program defined by the engineer. The reinforcement program is also used to generate an indication of the performance level of the module. Learning is automatically enabled whenever the performance of the module falls below a predefined threshold (e.g., 90% positive reinforcement signals).

Modules are allocated by the human user by means of behavioral decomposition. The user must decide:

- what type of behavior the module should implement;

- the sensory input to which the module has access;
- which motors the module controls;
- either pre-program the behavior, or design a Reinforcement Program and let it learn while the robot interacts with the environment;
- repeat this procedure for all modules in the architecture.

The pattern of connectivity among the modules and their individual activation networks are encoded on a binary string and evolved by a genetic algorithm. Evolution is incremental and operates on variable-length genotypes. Initially, a set of basic modules are defined by the engineer on the basis of available knowledge about the task requirements and of the characteristics of the robot shell.

An initial population of such controllers is evolved until an individual is generated that satisfies the task criteria. Individual modules with learning abilities can be separately trained before evolution and/or during evolution, depending on the task constraints. If the task constraints change, or if new hardware modules are added to the robot, it is possible to define new modules and increment the genotype length by including the new activation networks and all connections to previous modules. However, old parts of the genotype are masked so that they cannot be disrupted by the crossover and mutation operators.

This approach has been tested on two sequential tasks of incremental complexity (more details can be found in [33]). Initially, the task was that of developing a controller for the miniature mobile robot Khepera capable of moving the robot around the arena as much and as long as possible while periodically recharging the batteries at the recharging station (figure 2). The environment was a rectangular arena (40 cm by 60 cm) whose walls were covered with white paper. A 20 cm long metallic bar for battery charge was attached to one side of the arena and a 20-watt light bulb was positioned above it. The metallic bar could be used for recharging the robot batteries by using special contacts plugged on the robot.

The global task was decomposed in four simple behaviors: wander, obstacle avoidance, light following, and battery recharge, each corresponding to a module such as the one described above. Only the obstacle-avoidance module

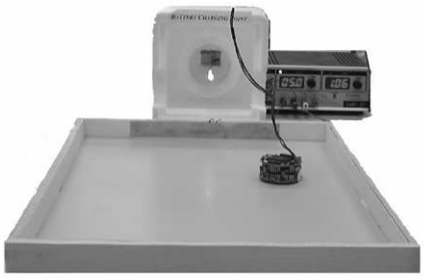

Figure 2: The Khepera robot is placed in the environment with a battery charger. The task is to keep moving in the environment and periodically recharge the batteries.

was adaptive so that it could cope with light changing conditions and changes in the properties of the sensors or of the motors. The behavior generator of the obstacle-avoidance module was an adaptive neural network mapping sensor activations into one of four possible motor actions (go forward, turn right, turn left, move backward). The reinforcement program punished increased sensor activations and rewarded decreased sensor activations. The learning algorithm was implemented as in [26]. The behavior generator of the remaining three modules was pre-programmed and fixed because it does not depend very much on external stimulation.

An initial population of 100 individuals was randomly created and evolved on the physical robot without any human intervention. Each individual, starting with a full battery, was tested for a maximum of 300 actions (a full battery allowed approximately 200 actions). Fitness points were accumulated at every step accordingto a function that encouraged straight motion and low sensor activation, as in [12]. Local learning in the obstacle-avoidance module was always active for all individuals. At each action a negative reinforcement was given (R = -1) if the activation of the proximity sensors increased, otherwise a positive reinforcement (R = 1) rewarded the performed action. Evolved controllers capable of recharging the battery could move around the environment for longer time and therefore accumulate more fitness scores than others. After 40 generations (approximately 2 days), several individuals in the population could optimally perform the task.

In a second stage, we added a number of small objects in the environment and equipped the Khepera with a gripper module (figure 3). The gripper module has two degrees of freedom: it can lift/lower the arm and open/close the gripper. An optical barrier between the two segments of the gripper provides sensory information on the presence of an object. The new extended behavior was that of collecting the highest number of objects and releasing them outside the arena, while maintaining the already evolved abilities of navigation and battery recharge.

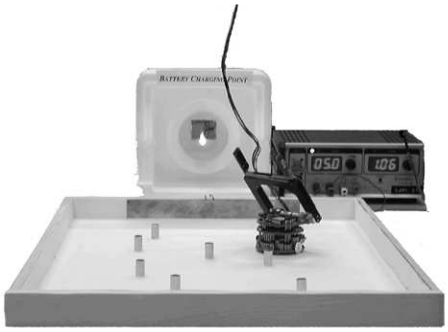

Figure 3: Small objects of roughly equal size are randomly distributed in the environment and the Khepera robot is equipped with a gripper module. The additional task is to find the objects, pick them up and release them outside the arena while maintaining the previous navigation and recharging abilities.

The new task was decomposed in two relatively-complex behaviors: object gripping and object releasing. The complexity came from the fact that each module must learn to discriminate between objects and walls. Both the object-gripping and the object-releasing modules learned autonomously to distinguish between objects and walls by lowering the gripper in front of the detected obstacle, observing whether the optical barrier between the arms detected an object, and associate the sensory pattern of infrared sensor activation with the corresponding category (figure 4).

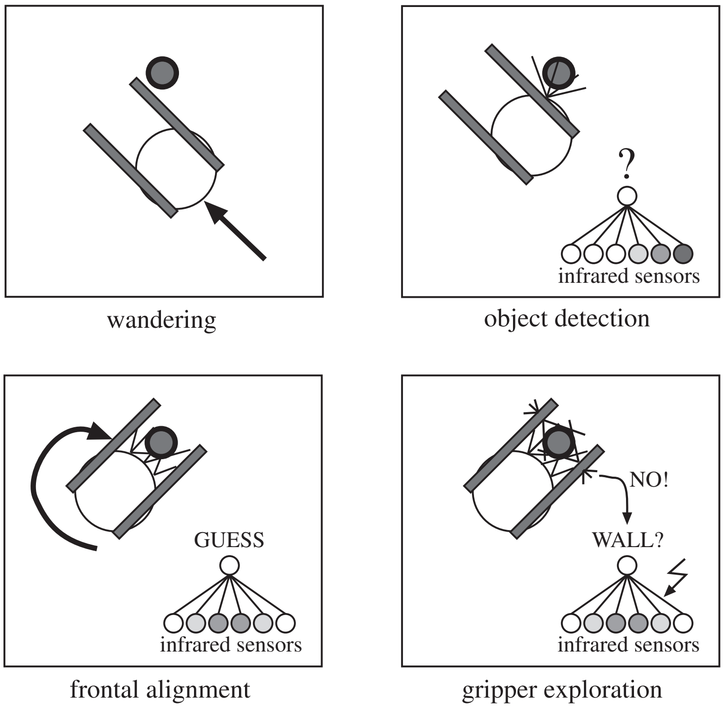

Figure 4: Self-supervised learning to discriminate objects by active exploration. Sequence is depicted clockwise from top left.

The new modules were fully connected to all previously existing modules and the genotype representation was augmented by including the strengths of the new inhibitory and excitatory links and the strengths of the synaptic values of the activation networks; previous parts of artificial the chromosomes were masked against mutation and crossover. The fitness function, which maintained the previous components, was extended by adding 0.5 points for every gripped object and 1.0 points if the object was correctly released outside the arena (see also [29] for a similar fitness definition).

After only 15 generations of incremental evolution, some individuals displayed the ability to explore the environment, discriminate between obeject and walls, picking up objects, releasing them outside the walls, and avoiding walls if no objects were being carried in the gripper; furthermore, they reliably and safely abandoned any current task and initiated a recharging procedure whenever the residual battery charge reached an autonomously determined low threshold.

The objects employed during evolutionary training had a diameter of 10 mm. In a third stage, we substituted them with larger objects with a diameter of 25 mm. Initially, the robot began misclassifying most of the new objects as walls. However, those few objects that were correctly picked up were sufficient to re-adapt the object modules to discriminate between new objects and walls (for more details, see [33]).

3.1 Discussion

The experiment described above empirically shows that an evolutionary and learning approach can be applied to a modular architecture based on behavioral decomposition to develop incrementally a complex task in interaction with the environment. The process is relatively constrained and it requires some knowledge from a human user in the specification of the module structures and of the fitness function. Nevertheless, this knowledge is much less than that required to develop a pre-defined program that does exactly the same task in the real environment. Furthermore, this methodology can autonomously re-adapts to minor changes in the environment or in the properties of the sensorimotor interface of the robot, and it can accomodate further behavioral skills while preserving older ones.

In this case, evolution here is used as an optimization process under relatively-well defined constraints. The problem is slightly different than standard function optimization because the experiences that the robot encounters is not fixed and distributed according to a normal distribution, as it is often the case for data analysis problems, but it depends on the robot actions and thus it changes as the robot develops better and new performances. Nevertheless, one cannot expect emergence of new phenomena, behaviors, and autonomous solution to new challenges.

As I will stress below, one of the key factors that distinguishes artificial evolution from evolutionary optimization is the amount of constraints put into the fitness function and into the architecture of the system. Consider one of our earlier experiments where we evolved a robot capable of developing a navigation and recharging behavior [12] functionally similar to that evolved in the first stage of the experiment described above. There, we evolved a recurrent neural network where the only constraint was that it only had five processing artificial neurons. Here, instead we put much knowledge about the environment in the type of architecture under development and we pre-programmed simple reactive behaviors, leaving to evolution the only task of co-ordinating module activations. In the former case, we observed a number of emergent solutions and phenomena that were unexpected and, under certain conditions, resembled the strategies used by rats to navigate in unknown environments. In the case described here, the evolved behavior was exactly the desired behavior, there were no unexpected and emergent properties, and the evolutionary process took only one fifth of the time to produce reliable solutions. These characteristics fit the requirements of many industrial applications, while offering a new and flexible approach for developing control architectures that rely on the adaptive interaction between the robot and its own environment.

4 Artificial Life and Robotics

Within an Artificial Life perspective, robots are seen as artificial organisms that autonomously develop solutions to cope with the challenges of the environment. Whereas the application-oriented approach described in the previous section explicitly imposes conditions and constraints in order to guarantee the desired result, Artificial Life attempts to understand the conditions and constraints under which certain behaviors emerge. Artificial Life approaches to robotics usually do not define strong constraints aimed at developing pre-specified competencies, but rather study the emergent phenomena that arise from certain initial conditions.

The notions of autonomy and intelligence here take a new meaning. From an application-oriented point of view, autonomy often means energetic autonomy or operation without external assistance. Therefore, a robot that runs on batteries according to a pre-defined program could be defined autonomous. In biological and artificial life, autonomy is instead related to the notion of self-sufficiency, the ability of an organism to develop solutions to environmental challenges in order to remain viable [25]. The two notions of autonomy are completely independent. From an Artificial Life point of view, a behaving robot can be autonomous even if it is tethered and its control program runs on a workstation.

In the opinion of the general public and often in the robotic literature, the intelligence of a machine is measured in terms of the number of automatic processes that are carried out without asking human intervention. For example, on new BMW cars you have *intelligent wipers* that start cleaning the windscreen as soon as it begins raining. Similarly, you have *intelligent robots* that can plan and execute an efficient trajectory from any starting position to a desired location using a model of the environment. In these cases, one can hardly define such machines as intelligent. Their performance and the complexity of their behavior simply shows the intelligence of the engineer who has developed them. In my view intelligence cannot be defined as a list of abilities, but as the very process of autonomous discover of solutions to environmental challenges. From this point of view, Artificial Life provides the settings and the epistemological background for studying and developing genuine artificial intelligence.

Accepting that autonomous adaptation to environmental challenges plays a central role in biological and artificial life implies that our notion of progress, as usually intended in machine learning, might not necessarily be valid. Several animal species often share the same ecological niche and their survival probability can be affected—more or less directly—other individuals. Although some theoretical biologists have assumed that co-evolution can sustain evolutionary progress, there is no empirical and theoretical evidence for it [19]. Co-evolving species might in fact drive one another along twisting pathways where each new solution is just good enough to counter-balance the current strategies implemented by the co-evolving species, but is not necessarily more complex than the solution used some generations earlier.

In other words, whereas in typical problems of evolutionary optimization the fitness landscape is usually static, in biological and artificial evolution the fitness landscape can change unpredictably. In the following sections, I will describe competitive co-evolution, an extreme case of co-evolutionary dynamic environments, and then present an overview of our investigations using co-evolving predator and prey robots.

5 The Red Queen of Evolution

In the simplest scenario of two co-evolving populations, fitness progress is achieved at disadvantage of the other population's fitness. Changes in one species might affect the selection pressure on the other species and, if the other species also responds by developing counter-adaptive features, the system might give rise to what biologists call a "co-evolutionary arms races" [9]. Although it is easy to point out several examples of such situation in nature (e.g., competition for limited food resources, host-parasite, predator-prey), it is more difficult to analyze and understand the importance and long-term effects of such "arms races" on the development of specific genetic traits and behaviors. From a computational point of view the question is whether the escalation in the development of counter-adaptive features by competing species does correspond to genuine progress in each individual species.

An interesting complication is given by the "Red Queen effect" (the Red Queen is a figure, invented by novelist Lewis Carroll, who was always running without making any advancement because the landscape was moving with her) whereby the fitness landscape of each population is continuously changed by the competing population [34] (figure 5, top). The Red Queen effect brings in two consequences. The first is that progress in one species could be eliminated by the competing species and the whole co-evolutionary process might result simply in random generation of different features. The second consequence is that, even if the two species might undergo true progress, the traditional fitness scores measured during evolution cannot reveal it to us because these values are obtained against different competitors (figure 5, bottom). Properly speaking, there is no fitness function in competitive co-evolution.

Theoretical models of competitive co-evolution (based on Lotka-Volterra equations) study how population density (i.e., the number of individuals) varies as a function of pre-defined abilities of the two competing species [28]. Therefore, these models cannot help us to predict whether artificial competitive co-evolution can be exploited for optimization purposes.

Despite these complications, the computational literature shows several examples where competitive co-evolution can generate powerful solutions to difficult problems. Hillis (1990) reported a significative improvement in the evolution of sorting programs when parasites (programs deciding the test condi-

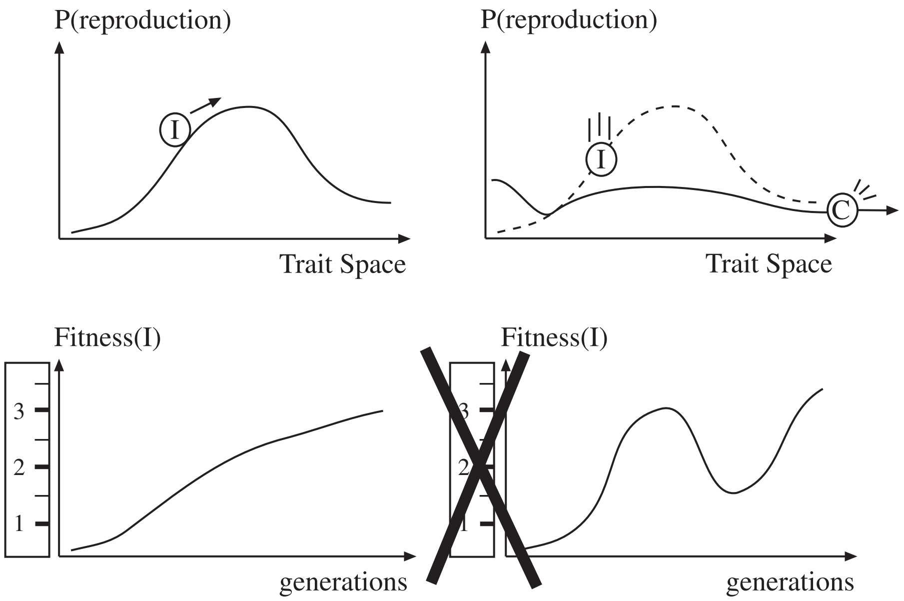

Figure 5: **Top Left**: Reproduction probability of a single species I under evolution in a static environment. **Bottom Left**: Evolutionary progress can be measured using fitness scores reported by species I during evolution. **Top Right**: Reproduction probability of species I under competitive co-evolution. **Bottom Right**: Fitness scores reported by species I depend on the strategies used by the competing species C. Fitness scores do not provide absolute measure of performance.

tions for the sorting programs) were co-evolved , and similar results were found by Angeline and Pollack (1993) on co-evolution of players for the Tic Tac Toe game. More recently, Rosin and Belew (1997) compared various co-evolutionary strategies for discovering robust solutions to complex games.

In the context of adaptive autonomous agents, Koza [21, 22] applied Genetic Programming to the co-evolution of pursuer-evader behaviors, Reynolds (1994) [30] observed in a similar scenario that co-evolving populations of pursuers and evaders display increasingly better strategies, and Sims used competitive co-evolution to develop his celebrated artificial creatures [32]. Cliff and Miller realised the potentiality of co-evolution of pursuit-evasion tactics in evolutionary robotics. In a series of papers, they described a 2D simulation of simple robots with evolvable "vision morphology" [27] and proposed a new set of performance and genetic measures in order to describe evolutionary progress which could not be otherwise tracked down due to the Red Queen Effect [7]. Recently, they described some results where simulated agents with evolved eye-morphologies could either evade or pursue their competitors from some hundred generations earlier and proposed some applications of this methodology in the entertainment industry [8].

One reason why competitive co-evolution might produce complex solutions, rather than simple noise, is that the ever-changing fitness landscape, caused by the struggle of each species to take profit of the competitors' weaknesses, might prevent stagnation of the two populations in local maxima. The other reason is that co-evolution could work as a sort of *intrinsically incremental evolution*. The hypothesis is that initially the fitness landscape for both species is rather flat. Therefore, both populations can easily climb up areas associated to higher valuesThe evolution of more specialized competencies makes the fitness landscape more rugged, but since the populations are already on local maxima, they can easily climb further up. After some generations, the fitness landscape might be so rugged that a randomly initialized population could not reach the peaks. If this is the case, competitive co-evolution might develop behavioral strategies that could otherwise not be discovered, and might do it faster. This incremental aspect of co-evolution might be exploited for evolving complex controllers for autonomous robots.

6 Competitive Co-Evolutionary Robotics

We have studied competitive co-evolution using two Khepera robots (for more details, see [15]), one of which (the *Predator*) was equipped with a vision module while the other (the *Prey*) had a maximum available speed set to twice that of the predator (figure 6). Both species were also provided with eight infrared proximity sensors (six on the front side and two on the back): a wall could be detected at a distance of 3 cm approx., but another robot could only be detected at half that distance.

The two species evolved within a square arena of size 47 x 47 cm with high white walls so that the predator could always see the prey (if within the visual

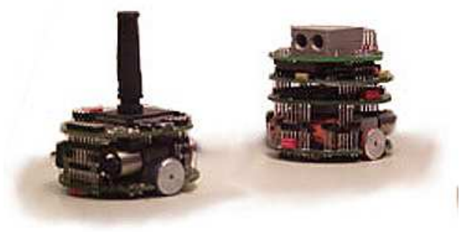

Figure 6: **Right**: The Predator is equipped with the vision module (1D-array of photoreceptors, visual angle of 36°). **Left**: The Prey has a black protuberance which can be detected by the predator everywhere in the environment, but its maximum speed is twice that of the predator. Both Predator and Prey are equipped with 8 infrared proximity sensors (max detection range was 3 cm in our environment).

angle) as a black spot on a white background.

The two robots were connected to a desktop workstation equipped with two serial ports through a twin aerial cable providing the robots with electric power and data communication to/from the workstation. The two cables ended up in two separate rotating contacts firmly attached to the far ends of a suspended thin bar. Both wires then converged into a single and thicker rotating contact at the center of the bar and ended up in the serial ports of the workstation (figure 7). The thick rotating contact allowed the bar to freely rotate around its center while the remaining two contacts allowed free rotations of the two robots. Attached under the bar was also a halogen lamp (20 W output) providing illumination over the arena.

Both robots were also fitted with a conductive metallic ring around their base so that they could detect when they hit each other. The vision module K213 of the predator consists of a 1D-array of 64 photoreceptors which provide a linear image composed of 64 pixels of 256 gray-levels each, subtending a view-angle of 36°. In the simple environment employed for these experiments, the prey looks like a valley whose width is proportional to the distance from the predator (figure 8, top) and its position indicates the relative position of the

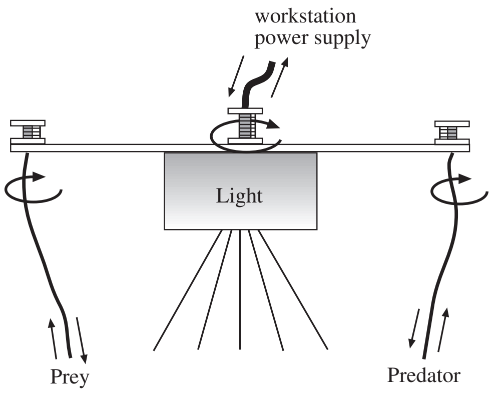

Figure 7: The suspended bar with the three rotating contacts and a white box casting light over the arena.

prey with respect to the predator.

The controllers of the two robots were two simple neural networks with recurrent connections at the output layer. This simple set up was chosen to keep low the number of free parameters under evolution so that we could later perform an analysis of the fitness landscape. The neural controller of the prey had eight input units connected to the eight infrared sensors and two motor units controlling the speed of each wheel. The neural controller of the predator was very similar, but its input layer had an additional set of five units that received information from the vision module. Each of these visual units, which covered approximately 13° and was inspired upon the complex cells found in the Lateral Geniculate Nucleus and cortex of mammals, responded whenever the visual projection of the prey fell within their receptive field. The maximum speed of the predator was set to twice that of the prey.

Two genetic algorithms were run in parallel, one for the predator and the other for the prey. The genetic algorithms were running on the workstation CPU, but each new neural controller was downloaded through the serial line into the microcontroller Motorola MC68331 of the two Khepera robots, which was sufficient to store the set of instructions and variables necessary to handle all input/output routines and neural states (figure 9). The speed of the input/output cycles was set to approximately 15 Hz for both prey and predator. Image acquisition and low-level visual preprocessing was handled by a private

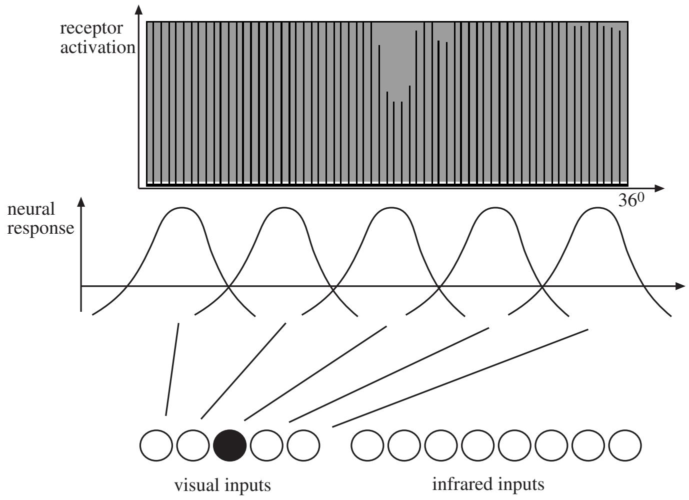

Figure 8: **Top**: A snapshot of the visual field of the predator looking at the prey. The heights of vertical bars represent the activations of corresponding photoreceptors. The black protuberance of the prey looks like a large valley. The small dip on the right corresponds to the cable. In standard illumination conditions, the image was refreshed at a rate of approximately 15 Hz. **Bottom**: Visual filtering with five *center/off surround/on* neurons. A neuron is maximally activated when the projection of the prey falls in its receptive field. The most active neuron is set to 1, all the remaining neurons are set to 0 in a Winner-take-All fashion.

68HC11 processor installed on the K213 vision turret. Upon receipt of a new controller, each robot began to move and the internal clock (a cycle counter) of the prey was reset to zero. A tournament ended either when the predator hit the prey or when 500 sensory motor cycles (corresponding to approximately 35 seconds) were performed by the prey without being hit by the predator. Upon termination, the prey sent back to the workstation CPU the value of the internal clock (ranging between 0 and 499) which was used as fitness value for both prey and predator (for the prey high values corresponded to high fitness, for the predator the other way around).

Two populations (one for the prey, the other for the predator) of 20 individuals each were co-evolved for 25 generations in approximately 40 hours of

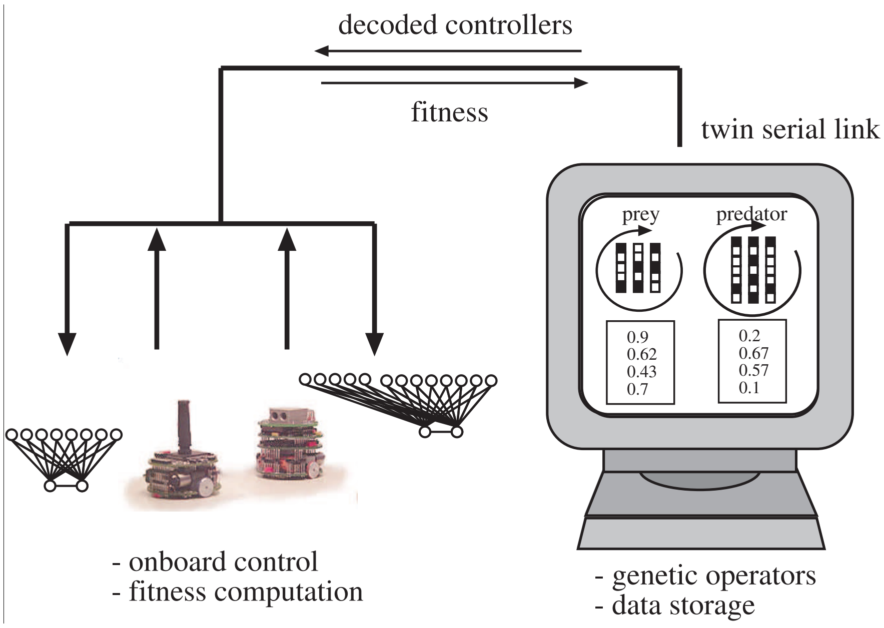

Figure 9: The genetic operators run on the main workstation, which also manages data storage and analysis; the neural controllers are automatically downloaded on the microcontrollers of the robots through the serial link. In the predator, a microprocessor on the vision module performs visual pre-processing and sends data at 15 Hz frequency to the main microcontroller.

continuous operation. Each individual was tested against the best competitors from the most recent 5 generations. Instantaneous fitness scores measured at each generation display oscillatory dynamics, as expected from the tightly-related dynamics and the Red Queen effect described above. A more detailed analysis of these data and of further analysis is given in [14, 15]. Here, I wish to point out that what these data tell us, especially the fitness scores of the best individuals, is that prey and predator rapidly (few that five generations) develop counter-strategies to defeat the competitors.

After only 15 generations, the two robots developed a variety of complex behaviors, such as those illustrated in figure 11. The prey display variations on two main strategies: fast motion around the arena and swift avoidance of the incoming predator. In the former case, the prey always moves around the

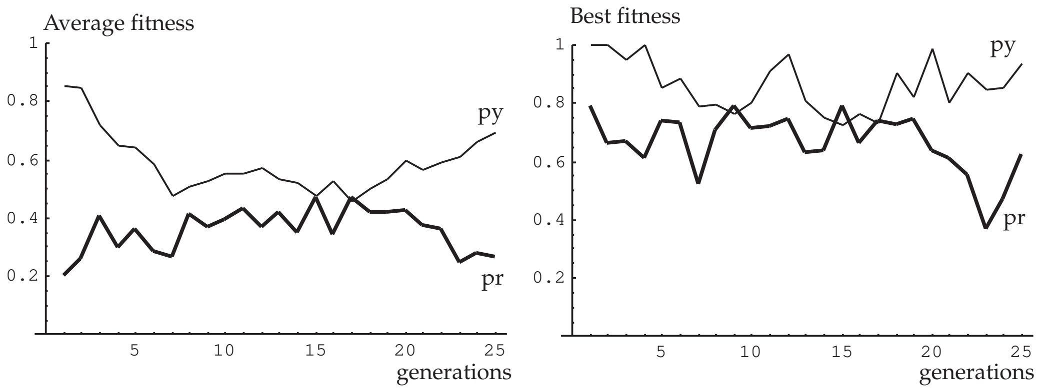

Figure 10: Co-evolutionary fitness measured on the real robots. *Left*: Average population fitness. *Right*: Fitness of the best individuals at each generation. `pr`=predator; `py`=prey.

arena, often following walls, and attempts whatever it finds on its way. Only predators capable of anticipating the prey's speed and direction of motion can capture it. However, the short-range response of the infrared sensors of the prey is such that sometimes the prey cannot avoid an incoming predator which is too small too reflect much infrared light. In the other case, the prey does not move waiting for the predator. As soon as it senses the predator, it quickly moves a few steps away; then, the predator performs a rotation on itself and it re-attacks, but the prey swiftly moves again a few steps away. This sort of dance ends when the predator happens by chance to attack the prey on the side where it has no sensors.

If one measures the average distance between predators and prey across generations, the data show that while prey attempt to maximize the distance from the predators, predators do *not* attempt to minimize the distance. In fact for both prey's strategies, best predators do not move straight after the prey (which would be a hopeless strategy since the predators are slower), but wait (rotating on themselves, for example) until the prey is at a proper distance and angle to attack (for quantitative data, see [15]). Distance is computed by observing how quickly the prey moves on the visual field, given that the predator and the prey usually move at a rather constant speed.

The Red Queen effect illustrated on the top of figure 5 is suspected to be the

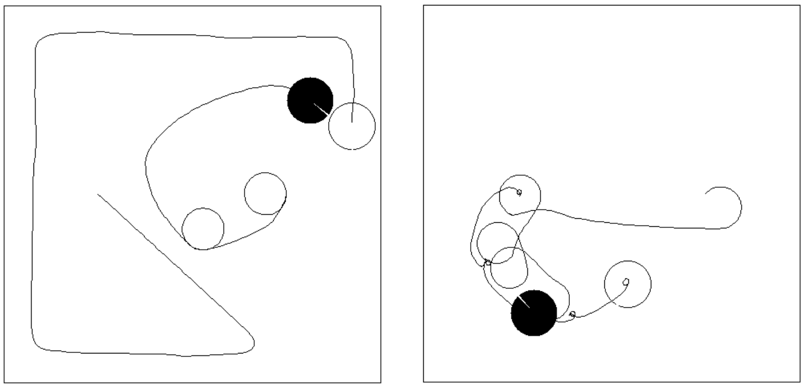

Figure 11: Examples of strategies developed by the robots. Black disk is the predator, white disk is the prey. Trajectories have been plotted running a tournament with simulated individuals who display the same behavioral strategies observed with the real robots.

main actor behind the dynamics, complexities, and computational advantages of competitive co-evolution, but how exactly it operates is not known. Capitalizing on the fact that our simple experiment with the robots displayed dynamics similar to those measured in experiments carried out in simulation, we exploited our computer CPUs to study how the fitness surface of one species is affected by the co-evolving competitor.

Given its shorter genotype length, we analyzed how the fitness surface of the prey was changed when confronted with the best predators recorded at different generations. The genotype of the prey was composed of 22 synapses encoded on 5 bits each. Assuming that the most significative bits are those coding the sign of the synapses, we are left with 22 bits.[1]. Each of the corresponding 4,194,304 prey was separately tested against the pre-recorded best predators of the first eight generations and against the best predator of generation 20, yelding a total of almost 40 million tournaments.

The fitness values were grouped into 100 bins of 4,194 values each (discarding remainders) and the average value of each bin was plotted on the graphs of figure 12 (a full picture can be found in [15]). The Red Queen effect is clearly

[1]The remaining 4 bits for each synapse were set at 0101, a pattern that represents the expected number of on/off bits per synapse and also codes for the average synaptic strength.

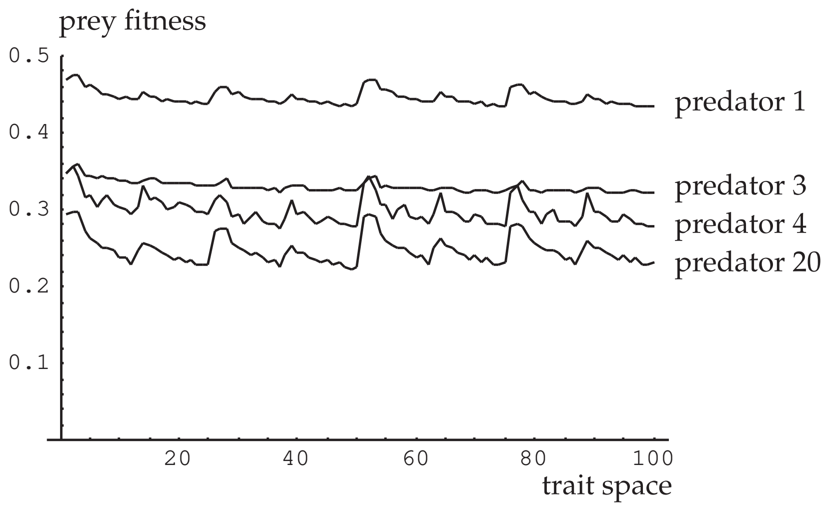

Figure 12: Fitness surface for the prey tested against the best predators from generation 1, 3, 4, and 20. Each data point is the average over the fitness values reported by 4,194 contiguous individuals.

visible in the temporary and periodic smoothing of the fitness landscape, as highlighted in figure 12. For example, the best predator of generation 3 causes a redistribution of the fitness values, stretching out the relative gain of some trait combinations with respect to others. This smoothing effect is always temporary and roughly alternates with recovery of a rough landscape.

It should be noticed that some regions corresponding to better fitness remain relatively better also during periods of stretching, whereas others are canceled out. That implies that individuals sitting on these latter regions would disappear from the population. If we view these regions as minima or brittle solutions, our data show the potentials of the Red Queen for optimization problems. Furthermore, it can be noticed that the steepness of the surface around the maxima becomes more accentuated along generations. If we assume that steeper regions are harder to climb, competitive co-evolution might facilitate progressive development of abilities that would be difficult to achieve in the scenario of a single species evolved in a static environment.

7 Discussion

The methodology and results obtained from these experiments are quite different from those obtained in the controlled experiments described in section 3. The main differences are to be found in the amount of structured knowledge imposed on the control architecture, on the definition of the fitness function, and on the characteristics of the fitness landscape. Also the background philosophy is quite different. Competitive co-evolution is not necessarily optimization in the sense of solution discovery for a pre-specified problem. The teleological interpretation of artificial evolution that underlies most research in genetic algorithms [2] and some research in evolutionary theory [19] leaves space to a more complex and richer scenario where robust solutions and true innovation can endogenously arise from simple interactions between parts of the (co-evolutionary) system.

To this extent, although the evolutionary approach can already be used for tasks that must comply with constraints and standards of some industrial applications, genuine innovations will spring out from artificial life approaches. An Artificial Life approach to Evolutionary Robotics can provide new insights, new tools, new methods and generate more genuine intelligent machines, in the sense that they autonomously and actively find solution to unpredicatable challenges put forward by the environment. Artificial Life implementations of robotics is also a healthy reminder that sooner or later a radical mind shift must take place in many industrial applications of robotics. Real environments cannot be characterized by mathematical models, pre-defined rules, precise measurements, and static landscape ready to be hillclimbed. Lessons from Artificial Life are already being incorporated in new robots for research and real-world applications.

8 Acknowledgements

The research work described here has been carried out in collaboration with Stefano Nolfi (CNR-Rome), Joseba Urzelai (LAMI-EPFL), and Francesco Mondada (K-Team S.A.). Some parts of this manuscript have been extracted from full reports to be appear elsewhere [33, 15]. The predator robot described in section 6 has been made available by K-Team SA. The author acknowledges support of the Swiss National Funds, grant nr. 21-49'174.96.

References

[1] P. J. Angeline and J. B. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 264–270, San Mateo, CA, 1993. Morgan Kaufmann.

[2] W. Atmar. Notes on the Simulation of Evolution. *IEEE Transactions on Neural Networks*, 5:130–148, 1993.

[3] L. B. Booker, D. E. Goldberg, and J.H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, 1989.

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

[5] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–59, 1991.

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

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

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

[9] R. Dawkins and J. R. Krebs. Arms races between and within species. *Proceedings of the Royal Society London B*, 205:489–511, 1979.

[10] M. Dorigo. Special issue on learning autonomous robots. *IEEE Transactions on Systems, Man and Cybernetics-Part B*, 26:361–364, 1993.

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

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

[13] D. Floreano and F. Mondada. Hardware Solutions for Evolutionary Robotics. In P. Husbands and J-A. Meyer, editors, *Proceedings of the first European Workshop on Evolutionary Robotics*. Springer Verlag, Berlin, 1998.

[14] D. Floreano and S. Nolfi. God Save the Red Queen! Competition in Co-evolutionary Robotics. In J. Koza, K. Deb, M. Dorigo, D. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Proceedings of the 2nd International Conference on Genetic Programming*, San Mateo, CA, 1997. Morgan Kaufmann.

[15] D. Floreano, S. Nolfi, and F. Mondada. Competitive Co-Evolutionary Robotics: From Theory to Practice. In R. Pfeifer, editor, *From Animals to Animats V: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1998.

[16] D. Floreano and J. I. Urzelai. Evolution and learning in autonomous robots. In D. Mange and M. Tomassini, editors, *Bio-Inspired Computing Systems*. PPUR, Lausanne, 1998.

[17] P. Gaussier. Special Issue on Animat Approach to Control Autonomous Robots interacting with an unknown world. *Robotics and Autonomous Systems*, 16, 1995.

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

[19] S. J. Gould and R. C. Lewontin. The spandrels of San Marco and the Panglossian paradigm: a critique of the adaptationist program. *Proceedings of the Royal Society London, B*, 205:581–598, 1979.

[20] W. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[21] J. R. Koza. Evolution and co-evolution of computer programs to control independently-acting agents. In J.-A. Meyer and S. Wilson, editors, *From Animals to Animats. Proceedings of the First International Conference on Simulation of Adaptive Behavior*. MIT Press, Cambridge, MA, 1991.

[22] J. R. Koza. *Genetic programming: On the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, 1992.

[23] B. Kroese. Special Issue on Reinforcement Learning and Robotics. *Robotics and Autonomous Systems*, 15, 1995.

[24] M. Matarić. Special Issue on Complete Agent Learning in Complex Environments. *Adaptive Behavior*, in press, 1997.

[25] D. J. McFarland and T. Boesser. *Intelligent Behavior in Animals and Robots*. MIT Press/Bradford Books, Cambridge, MA, 1993.

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

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

[28] J. D. Murray. *Mathematical Biology*. Springer Verlag, Berlin, 1993. Second, Corrected Edition.

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

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

[31] C. Rosin and R. Belew. New methods for competitive co-evolution. *Evolutionary Computation*, 5(1):1–29, 1997.

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

[33] J. Urzelai, D. Floreano, M. Dorigo, and M. Colombetti. Incremental robot shaping. In J. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, San Francisco, CA, 1998. Morgan Kaufmann.

[34] L. van Valen. A new evolutionary law. *Evolution Theory*, 1:1–30, 1973.