Adaptive **Behavior**

Evolving Controllers for Real Robots: A Survey of the Literature

Joanne Walker, Simon Garrett, Myra Wilson
Department of Computer Science, University of Wales

For many years, researchers in the field of mobile robotics have been investigating the use of genetic and evolutionary computation (GEC) to aid the development of mobile robot controllers. Alongside the fundamental choices of the GEC mechanism and its operators, which apply to both simulated and physical evolutionary robotics, other issues have emerged which are specific to the application of GEC to physical mobile robotics. This article presents a survey of recent methods in GEC-developed mobile robot controllers, focusing on those methods that include a physical robot at some point in the learning loop. It simultaneously relates each of these methods to a *framework* of two orthogonal issues: the use of a simulated and/or a physical robot, and the use of finite, training phase evolution prior to a task and/or lifelong adaptation by evolution during a task. A list of *evaluation criteria* are presented and each of the surveyed methods are compared to them. Analyses of the framework and evaluation criteria suggest several possibilities; however, there appear to be particular advantages in combining simulated, training phase evolution (TPE) with lifelong adaptation by evolution (LAE) on a physical robot.

Keywords evolutionary robotics · physical robots · simulation · training · lifelong adaptation by evolution.

1 Introduction

1.1 Motivation

One of the major issues being addressed in robotics is that of training mobile robots to perform a task without external supervision or help.[1] One response to this challenge, which has received considerable attention, is the use of genetic and evolutionary computation (GEC) (Nolfi & Floreano, 2000; Hornby et al., 2000; Harvey, 1997; Nordin & Banzhaf, 1995). A well-known dichotomy has appeared in this research over the last few years between those who evolve GEC controllers using *simulated* robots (Jakobi, Husbands, & Harvey, 1995; Bongard, 2002) and those who use *physical* robots (Floreano & Mondada, 1994; Watson, Ficici, & Pollack, 1999):

- Simulated robots, that exist in simulations of the world, are used on the basis that such an approach is usually less expensive (no robot hardware, or damage to it caused by experimentation), can be faster, and allows the researcher to concentrate on developing the control method rather than the engineering issues that often surface with physical robots.
- Physical robots in the real world are used on the basis that "the world is its own best model"

Correspondence to: J. Walker, Department of Computer Science, University of Wales, Aberystwyth SY23 3DB, Wales, UK.
E-mail: jnw@aber.ac.uk,
Tel.: +01-970-621787, *Fax*: +01-970-622455.

(2003), Vol 11(3): 179–203.
[1059–7123 (200309) 11:3; 179–203; 040092]

(Brooks, 1986) and therefore simulation and off-line training are not only unnecessary, they can actually be misleading since no simulation can fully match real-world complexity (Mataric & Cliff, 1996).

Simulations of the robot and its environment have often been used exclusively during a training period for a number of practical and strategic reasons (Mataric & Cliff, 1996), but mainly because the time taken to run any sort of GEC on a physical robot is generally prohibitive. The practical advantages of using evolution in simulation have encouraged a number of researchers to investigate ways of improving the accuracy of robot simulators, so that smooth transfer from simulation to reality can take place: for instance the inclusion of the correct amount of noise within the simulation has been found to be significant (Jakobi et al., 1995; Miglino, Lund, & Nolfi, 1995b). However, there has also been a significant amount of work which investigates the use of evolutionary algorithms solely on physical robots, with no prior training in a simulated environment (such as Nehmzow, 2002, and Floreano, Nolfi, & Mondada, 1998). These choices highlight another distinction between:

- The development by GEC of a robot controller during a finite training phase that terminates before the robot is applied to a task. The controller is not adapted during the task. We will call this *training phase evolution* (TPE).
- The development of a controller that is adapted by GEC throughout the robot's task. We will call this *lifelong adaptation by evolution* (LAE).

Most evolutionary robotics work has used evolution exclusively during the training phase of the robot controller. The evolutionary algorithm is used to adapt the robot's controller to improve the robot's ability to perform its task in its environment. At the end of this training period the controller is used for the task for which it was designed, and during the task no further adaptation takes place. An alternative approach is for evolution and adaptation to occur during the task of the robot. This approach is not so prominent but shows promise. We note that TPE is *not* a subset of LAE because TPE occurs before the true task begins (even if it uses examples of the task as part of the training process), and LAE occurs during a task. A literature review is presented, structured by these two, orthogonal issues—simulation versus real robots, and TPE versus LAE—and possible new avenues of research are examined that are suggested by investigating this formalism.

Since this review focuses on the use of evolution in real robots, there is no in-depth discussion of methods that use evolution (whether TPE or LAE) only in a simulated setting. It has been shown that such approaches can easily evolve solutions that are adaptations to features of the simulation that are not present in the physical world (Brooks, 1992), and since the question addressed here is "*what is the best way to use evolution to build controllers for physical robots?*," they also are not our concern. For an excellent, if slightly dated, review of work in this area see Mataric & Cliff (1996). Finally, it is acknowledged that evolution is not the only mechanism for adaptation that has been used in robotics, but it is the sole mechanism under investigation here.

The main contributions of this article are:

- A new review of research in evolutionary robotics.
- A proposed framework for categorizing evolutionary robotics is presented that serves to highlight aspects of this form of research that are currently receiving little or no attention.
- A proposed set of criteria is suggested that may be used to assess the value of any GEC controller method.
- An assessment of the relative merits of using simulated and physical robots in the development of robot controllers.
- An assessment of the relative merits of the TPE and LAE methods of developing robot controllers.
- Suggestions for, and illustrations of, *combined* TPE and LAE methods that use the advantages of simulation before porting to a physical robot.

1.2 Structure of the Article

The article is structured as follows: Section 2 gives a brief overview of the major GEC algorithms and an explanation of how they function. Section 3 first introduces the framework used to relate the various types of evolutionary robotics discussed in this article, and then defines the evaluation criteria used to assess the

value of that research, with an explanation of the criteria chosen. The literature survey itself is split into three sections with Section 4 describing TPE methods, Section 5 examining LAE methods, and Section 6 focusing on the few projects that have combined TPE and LAE into a single method. Section 7 then suggests some new directions and draws some conclusions about their viability.

2 Background: Genetic and Evolutionary Methods

There are four main, interrelated topics in genetic and evolutionary computation: genetic algorithms (GAs), evolutionary strategies (ES), genetic programming (GP), and evolutionary programming (EP). All GEC methods are based (if somewhat loosely) on the concepts of genetics and evolutionary selection, and their terminology reflects this. Historically, the clear majority of evolutionary robotics has used GAs. More recently there is an understanding that these methods represent areas within the continuum of GEC methods, with less distinction being made between the different types.

2.1 Genetic Algorithms

GAs are usually attributed to Holland's work in the mid-1970s (Holland, 1975); they can be defined as follows. Consider a population, $p_i \in P$, of possible solutions to (or optimizations of) a problem. Each p_i is known as a *chromosome*, or *genotype* containing a vector of values $\vec{v}$, and in some cases some other elements too. Traditionally each $v_i \in \vec{v}$, are chosen from a binary alphabet, {0, 1}, where each bit, or group of bits, encodes part of the genotypes's proposed solution. Other types of encoding include integer and real-valued v_i elements. These genotypes are tested by a fitness function, which assesses how good the potential solution actually is, and manipulated by a number of operators—usually selection, crossover, and mutation. A complete iteration of fitness function evaluation and application of operators to the whole population, known as a generation, probabilistically results in an increase in fitness of the population, as a whole. Subsequent generations tend to result in further increases in population fitness, until a predefined number of generations have occurred or some fitness level is reached. More detailed descriptions can be found (Mitchell, 1998; Goldberg, 1989; Holland, 1975, 1993).

2.2 Evolution Strategies

ESs were first introduced in German by Rechenberg (1965), and the ideas were revived in the early 1970s (Rechenberg, 1973; Schwefel, 1977). Both are later cited by Back & Schwefel (1993), written in English. They were very simple algorithms, employing a similar representation and use of operators to the GA, but with a single parent which produced one offspring per generation by mutation alone. The offspring would then replace its parent in the next generation, if it had a better fitness. Moreover, the mutation step size is defined by a Gaussian-distributed random number, with a mean of zero and a standard deviation that is encoded on a different part of the chromosome or genotype. The standard deviation is also mutated in each generation. This allows the ES to be self-optimizing in its performance. Unlike most GA approaches, ESs almost always use genotypes of *real numbers*.

2.3 Genetic Programming

Genetic programming is a relatively new evolutionary method that was developed by Koza (1992). To solve problems, GP evolves whole *computer programs*, in the form of trees, not just vectors of values. Koza's GP uses Lisp S-expression tree structures. The inner nodes of these trees represent functions of the program, and the leaf nodes represent variables and constants to the functions.

As with GAs and ESs, there is population of solutions (in this case a collection of trees), which are manipulated in each generation of evolution by crossover and mutation to form new trees. Like a GA, these new trees are then tested for fitness at each generation. GP can be seen as a method for hypothesis search, where the most fit solution is the smallest tree that correctly covers a set of input data, or (as in the context of this paper) a method for evolving programs such as robot controllers.

2.4 Evolutionary Programming

EP was developed by Fogel, Owens, and Walsh (1966) in the mid-1960s. In this initial work candidate solutions were represented as finite state machines (FSMs)

that were evolved by mutation and selection only. Crossover is not usually used in EP. A finite state machine is a directed graph of states, with transition from one state to another causing a symbol to be emitted. This sequential circuit takes a finite set of inputs, that define the next state, and the output is a logical combination of those inputs and the current state of the FSM. A significant factor in modern EP is that the representation structure is designed for the specific problem domain, for instance real-valued vectors, graphs, ordered lists, and trees may all be used in different domains (Spears, Jong, Back, Fogel, & de Garis, 1993).

2.5 The Problem of GEC in Robotics

Having seen the variety of GEC methods, there is one common issue that is particularly important for evolving on physical robots: the issue of how to initialize the GEC method's population. A random first generation genotype may lead to the robot being made to act in a manner that would cause it to damage itself, and it may require significant time before useful behaviors are seen; however, if the first generation is seeded with hand-designed genotypes, the task of evolution may become minimal, diminishing its usefulness. After analysis of previous methods, a middle way will be suggested in Section 6, which helps to mitigate both these problems.

3 Defining a Method of Analysis

3.1 A Framework for Existing Evolutionary Robotics Methods

Figure 1 puts the extant research in evolutionary robotics into a framework that will be used throughout this paper. Methods for the development of robot controllers are first divided into two groups: one in which evolution takes place exclusively in a training phase (the upper subtree), and the other in which evolution continues throughout the lifetime of the robot (the lower subtree). Within these groups, the existing types of simulated versus physical robotics are shown. Possibilities that have not yet been explored in existing research are not shown in Figure 1, but are discussed in Section 7.1 and shown in Tables 1 and 2.

Training phase evolution (TPE): Existing research that evolves a robot controller during a finite training

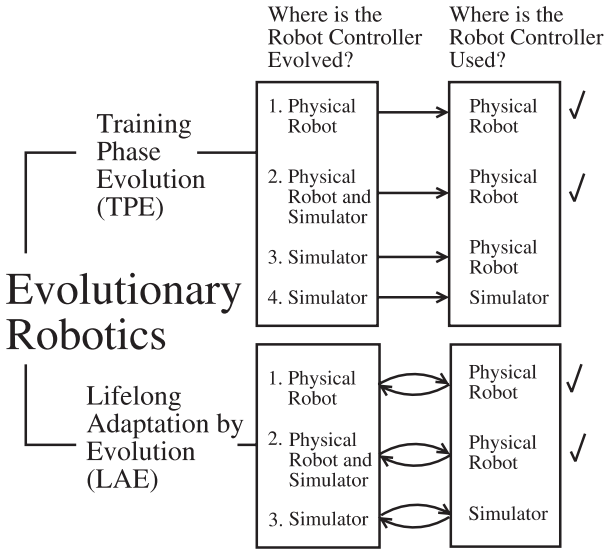

Figure 1 A suggested framework for the various types of evolutionary robotics. The sections marked with ticks are those being reviewed in detail in this paper.

phase can be subdivided into four approaches: (i) development and application take place solely on a physical robot; (ii) development occurs on a physical robot in the real world *and* in simulation, before it is fully ported to the physical robot; (iii) development takes place in simulation alone and the resulting controller is applied to a physical robot, and (iv) development and application both take place in a simulated world. Since, as stated above, the focus here controllers that are evolved on physical robots at some point, points (iii) and (iv) will not be examined further (see Section 1.1 for the reasoning on this point).

Lifelong adaptation by evolution (LAE): Existing research that evolves a robot's controller throughout its lifetime can also be subdivided, but in a slightly different manner. The difference is due to the iterative nature of the adaptation, since the learning loop has been closed and experience now feeds back to aid adaptation. There are three cases identified from the literature: (i) the sensory data from a physical robot are continually used in the evolution of the physical robot's controller; (ii) the physical robot's controller is evolved from physical and simulated sensory data, aggregated in some manner and the controller controls a physical robot, and (iii) the evolution and application of the controller both occur in simulation. Again,

work of type (iii) will not be further considered because at no point is the robot controller developed on a physical robot.

3.2 Criteria For Assessing the Usefulness of the Approaches

The following set of evaluation criteria, which have been distilled from a review of the literature, are used to assess approaches that evolve controllers for physical robotics. In general, these criteria are concerned with assessing how well each approach is able to provide a worthwhile alternative to design by hand, which is the challenge posed by Mataric and Cliff (1996). The criteria are split into two sections: criteria for good TPE methods, and criteria for good LAE methods. They will be applied to each TPE and LAE method.

3.2.1 Criteria for Training Phase Evolution The criteria for assessing the usefulness of TPE methods are as follows:

Time required for training: The time required for training must be as short as possible, and in any case not be prohibitive. The longer the training period, the less valuable it is compared to the hand-design of a robot controller. This issue has been discussed by many researchers, including Mataric & Cliff (1996), and Brooks (1992).

Generality from the training phase: The training phase may be carried out in an environment that approximates the world in which the robot will eventually carry out its task; however, no non-trivial environments are entirely regular, and if robots are to be useful in unconstrained surroundings then their controllers will need to be robust and general enough to make control decisions in circumstances not encountered during the training phase.

Accuracy and repeatability: The evolved controller must be able to accurately repeat its training, so that a given set of sensory inputs will reliably elicit the same appropriate response. This issue may be in conflict the criterion of generality above, as discussed by Mataric and Cliff (1996).

3.2.2 Criteria for Lifelong Adaptation by Evolution The criteria for assessing the usefulness of LAE methods are as follows:

Adaptation in real time: The LAE method must be fast enough to adapt to a changing environment. In a dynamic world, if a robot takes too long to adapt, the environment may have changed again before it can establish a fit response.

Overall improvement in performance: As well as the ability to adapt to punctuated changes in the environment, seen as recovery from short-term dips in performance, the controller should also be able to show an overall increase in performance throughout its lifetime, as it evolves to slower changes in the environment and adapts to more general control issues.

Interference of the evolutionary process in the robot's task: The logistics of implementing the GEC algorithm should not unduly interfere with the robot's task. For example, the use of a computer workstation to host the GEC, requires information to be transferred to and from the robot during its lifetime. This may require a pause in activity (for wireless transmission), the use of a cable tether that can affect the motion of the robot, or even physical docking with the computer. If this occurs too frequently the robot's task will be interrupted and the robot's task performance may degrade, but if it occurs too infrequently the benefits of evolution may be lost. Similar issues arise with the use of teams of robots that must communicate to transfer genetic material.

4 A Survey of TPE Methods

In order to clearly present the many approaches to the two parts of TPE being discussed here (parts 1 and 2 on the TPE branch of Figure 1, denoted TPE:1 and TPE:2), their methods are presented in related groups as follows:

Training on Physical Robots (TPE:1)

- Early work.
- Evolution and shaping.
- Evolving fuzzy rules for robot control.
- Walking in legged robots.
- Active vision.
- Non-GA GEC algorithms, namely GP and EP.

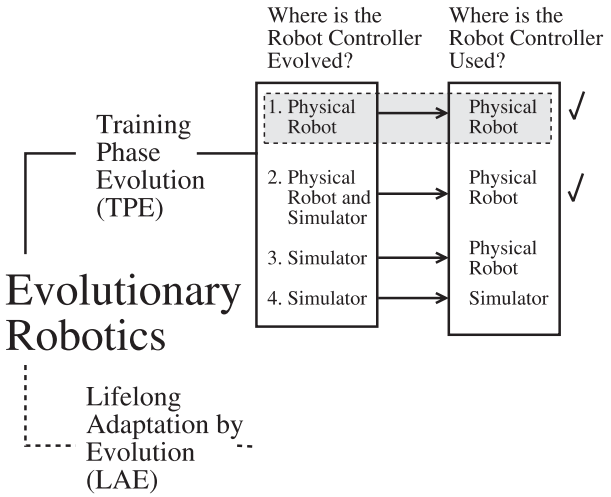

Figure 2 The position of Section 4.1 within the structure of this article, as defined in Figure 1.

Training on a Mixture of Simulated and Physical Robots (TPE:2)

- Simulation followed by fine tuning on a physical robot.
- Interleaving simulation and physical robots.

4.1 The "Physical Robot → Physical Robot" Form of TPE (TPE:1)

This section examines the application of robot-evolved controllers to a physical robot, where no use is made of simulation. Figure 2 highlights where this type of project fits into the overall structure defined in Section 3.1.

4.1.1 TPE:1—Early Work Until the early 1990s, all work in evolutionary robotics had used simulation for evolution, although some workers had tested their work on physical robots (Grefenstette, Ramsey, & Schultz, 1990; Jakobi, 1994; Jakobi et al., 1995; Gallagher, Beer, Espenschied, & Quinn, 1994).

One of the earliest attempts at carrying out evolution entirely on a real robot was made by Floreano and Mondada. They reported the successful evolution of a neural network for simple navigation and obstacle avoidance (Floreano & Mondada, 1994) on a Khepera robot, a robot widely used in evolutionary robotics research (Mondada, Franzi, & Ienne, 1993). A standard GA was used to evolve genotypes made up of floating point numbers that formed the weights and thresholds of the robot's neural network. This method was later applied to the evolution of the more complex homing behavior (Floreano & Mondada, 1996b), in which a robot learned to return to a light source that represented a battery recharge station. The experiments were successful, with good solutions being found in few generations.

Time required for training: Although all the experiments carried out by Floreano and Mondada produced successful results, the time taken to evolve the controllers was very large. The homing behavior took 10 days of continuous evolution, with the physical testing of solutions on the robot being by far the most time-consuming factor. The genotypes were tested serially on a single robot. Given the length of time required, it raises the question of whether it would not be faster to hand-design the controllers.

Generality of the controller: Nevertheless, the robustness of the results was encouraging. As part of the training phase of the grasping behavior experiments, the robot was first presented with a simple task and evolved solutions to it, and then it was given a harder task—some of the cylinders were removed so that they were harder to find—and remained able to use the previously evolved abilities to continue grasping cylinders. Although a drop in fitness was recorded, the GA soon found good solutions. Once training was complete the robot performed its task. Further experiments to test the final evolved behaviors under both easy and hard conditions would have been useful, to show how well the final behaviors generalized and how accurate they were.

Accuracy and repeatability: The evolved robots were able to accurately carry out their task after training. However, no report is made of how repeatable the behavior of the robots was after training.

4.1.2 TPE:1—Evolution and Shaping Colombetti and Dorigo report a system they call ALECSYS that used a form of evolution called learning classifier system (LCS) and a directed learning method called shaping to learn rules for carrying out tasks (Colombetti & Dorigo, 1992; Dorigo, 1995). LCSs use a GA to optimize rules, known as classifiers. An LCS is composed of three parts (Booker, Goldberg, & Holland, 1989): (i) a GA which finds new rules to add to a knowledge base; (ii) a performance system that controls the

behavior of a robot using rules, and (iii) an apportionment of credit system which evaluates the rules used by the GA and the performance system. Shaping is an approach to learning that uses a human trainer whose role is to direct the learning process, in this case by presenting increasingly complex learning goals over time until the final, complex goal is reached.

The ALECSYS experiments were performed on a robot known as AutonoMouse. Its task was to follow a moving light source. First the robot learned to move towards a stationary light by wandering around randomly, and being rewarded when it approached the light. In the next phase the system was specifically presented with situations it had not learned in the first phase, for example if the light was not easily accessed, and trained further. In the final phase the robot learned to move towards a moving light source.

Time required for training: Colombetti and Dorigo concluded that shaping significantly speeded up the learning process (although the actual amount of real time taken is not reported) as the system can be deliberately pointed in the right direction. This is a form of directed search. The aim is to prune unprofitable avenues for learning and to cut down the search space. However, this means that a trainer must be present and alert at all stages of the learning process.

Generality of the controller: Further experiments looked at the effect of altering the robot to see how quickly the system evolved to cope. For example in one experiment an "eye" (i.e., a light sensor) was removed. This is similar to the way in which Floreano and Mondada changed the robot's environment in their work. As with that work, ALECSYS recovered well in these situations. Again, once this training was complete, there were no further experiments to explore how well the system continues to operate under varying conditions.

Accuracy and repeatability: The evolved robots were able to accurately carry out their task after training. However, as in the work reported in the previous section, no report is made of how repeatable the behavior of the robots was after training.

4.1.3 TPE:1—Evolving Fuzzy Rules for Robot Control Matellan and others present a GA that evolves a fuzzy controller for a Khepera robot, which was required to navigate and avoid obstacles (Matellan, Molina, Sanz, & Fernandez, 1995; Matellan, Fernandez, & Molina, 1998). Fuzzy controllers use fuzzy rules that take into account the inaccuracy of human expressions such as "the wall is *quite* far away." A fuzzy rule might be expressed, "if the obstacle is quite near, then move away fairly fast." A given sensor reading maps on to a fuzzy subset (such as "quite near"), the fuzzy rules fire, and the result is *defuzzified* to give a real value for an actuator.

In their project Matellan et al. evolved fuzzy rules on a workstation and then downloaded each genotype onto the robot for testing. The resulting fitness was fed back into the GA for computation of the next generation. The findings were encouraging in that the approach found increasingly good solutions over successive generations. However, the controllers produced were found to be very similar to hand-designed ones and the evolutionary process was lengthy.

Time required for training: 100 individuals were tested on the Khepera over 100 generations, with each genotype controlling the robot for 20 s, totaling at least 55 h of continuous evolution time (excluding robot failures and other incidents); almost certainly longer than it would take to design such rules by hand.

Generality of the controller; accuracy and repeatability: No reports were made of how well the controller was able to generalize to new environments after training, nor about how repeatable the results were. In terms of accuracy the controllers were able to carry out their task, and as the solutions were similar to hand designed ones it is likely that their performance was also similar.

4.1.4 TPE:1—Walking in Legged Robots The development of efficient gaits for legged robots is a problem that has occupied many researchers, and GECs (mostly GAs) have become a popular approach. Here two example projects are discussed, one that evolves a quadruped gait and one that evolves a hexapod gait. A group of researchers working with the Sony quadruped, AIBO, have reported success using undefined GEC to evolve fast gaits (Hornby, Fujita, Takamura, Yamamoto, & Hanagata, 1999; Hornby et al., 2000). In early experiments, gaits were evolved on flat carpet, but were found not to generalize well to new

surfaces; subsequent experiments therefore used an uneven floor surface during evolution. This worked well but it was found that the experimenters had to be careful to get the level of unevenness right—the floor needed to be uneven enough to make the results robust, but not so rough as to make robot fall over, or to otherwise ruin the experimental procedure.

Time required for training: In these experiments each run of 500 generations took about 25 h to complete. For this reason the workers suggest using physical robots only when building a simulation would be too difficult (and therefore time-consuming in itself), and when fitness evaluation is fast, which is rarely the case with physical robots.

Generality of the controller: It was found that once uneven floor surfaces were used during evolution, the resulting gaits generalized well to new surfaces such as carpet and wood, and in addition they were faster than hand-designed gaits.

Accuracy and repeatability: The robot was able to walk accurately, but again there is no report of any experiments that considered the repeatability of the evolved behavior.

Incremental learning was employed in the training phase by Lewis, Fagg, and Solidum (1992) for gait-learning in a hexapod robot. A simple task was learned first, that of moving a single leg, followed by forming coordination between legs to evolve the walk itself. The genotypes were tested on the physical robot, but evaluated by hand, which is unusual because of the subjective nature of this approach, as well as the level of attention required by the evaluator.

The gait that evolved was a tripod gait, where the left-front and left-back legs move with the right-middle leg and vice versa. Also, a surprising finding was that the robot walked backwards more efficiently than forwards in the individuals produced by the GA. A relatively small number of generations were needed to produce good gaits, with a population of just ten individuals. It is useful to note that some of the best solutions—ones that had the robot walking backwards—were unlikely to have been hand-designed, as this was unexpected by the experimenters.

Time required for training: As a small number of individuals were used in each generation the time to evolve would have been smaller than similar projects. This small population size was probably made possible by the fact that the experimenter evaluated the individuals by hand although this may have led to some increase in evaluation time.

Generality of the controller; Accuracy and Repeatability: The robot was able to walk successfully after training, but there is no report of how repeatable this behavior was, nor how generalizable.

4.1.5 TPE:1—Active Vision Active vision, or active perception, is the use of motor actions to find sensory patterns that are easy to discriminate—see Bajcsy (1988) in Nolfi & Floreano (2000)—so that the so-called *perceptual aliasing problem* can be solved. For mobile robot vision, this means that when a number of different objects look identical, from a given position, the robot can move to another viewing position in order to disambiguate the objects. The active vision system, introduced by Kato & Floreano (2001), was implemented on a physical robot with a very simple task—moving around an arena whilst not hitting the walls (Marocco & Floreano, 2002). The active vision system took information from a very small part of the whole field of vision (48 by 48 pixels). This small area of focus they called the retina. The retina could be moved around the image, and zoomed in and out. A neural network was evolved on a Koala robot, equipped with a camera. The neural network controlled the pan and tilt of the camera and the motor speeds.

The best resulting genotypes were successful in avoiding obstacles, in this case walls. They used edge detection to recognize the meeting of the arena wall and the floor, and "visual looming," or the correlation between the size of the white wall in the camera view and the speed of motors moving the robot. The result was the ability to perform wall-avoidance.

Time required for training: A population of 40 individuals was evolved for just 15 generations, and because of this small population size and the number of generations, training took only ~1.5 h, much less than many other projects.

Generality of the controller: No experiments are reported which look at the issue generality after train-

ing. It would have been interesting to see if further training would be necessary before the robot could perform the same behavior in a different environment, and if so, how much further training.

Accuracy and repeatability: The authors report that although the evolved robot was able to satisfactorily carry out its task, it was not as good as a simulated robot in training.

4.1.6 TPE:1—Non-GA GEC Algorithms In evolutionary robotics, GP is a common alternative to a GA, and was used by Nordin et al. for the evolution of various controllers for a Khepera robot. The controllers took the form of computer programs that were manipulated by the GP and tested on a robot. The first experiments evolved a typical obstacle-avoidance behavior (Nordin & Banzhaf, 1995).

The GP was used to evolve machine code, which meant that the process was memory efficient, so that it could take place entirely on physical robots. The fitness function was also very simple, and was based on abstractions of "pleasure" and "pain," such that high values from the IR sensors (indicating close proximity to an obstacle) produced pain, and high, similar motor speeds (indicating fast forward movement) gave pleasure. In order to speed up the evolutionary process, just four individuals from each generation were tested on the robot and subsequently manipulated by the GP. In addition, each test run was kept short.

A number of more complex behaviors were then successfully evolved using this technique, including following moving objects (Banzhaf, Nordin, & Olmer, 1997) and action selection strategies (Olmer, Nordin, & Banzhaf, 1996).

A later version of the system was presented that included a memory of past experiences. This approach was used to learn obstacle avoidance again (Nordin & Banzhaf, 1997), and later wall-following (Nordin et al., 1998). The addition of a memory significantly improved the learning process, with perfect obstacle-avoidance being learned on average in 50 generations, and the more difficult task of wall-following being perfectly learned, on average, in 150 generations.

Time Required for Training: The measures taken to speed up the process meant that more generations could be produced in a short time, and it was found that the system successfully evolved obstacle-avoidance behavior in just 40–60 min (equivalent to 200–300 generations).

Generality of the controller: The controllers evolved for obstacle avoidance were put into new environments after training to test for robust generalization, and proved to perform well.

Accuracy and repeatability: The evolved robot was able to carry out the tasks very accurately. Although tests were done to see how repeatable the evolutionary process was in terms of the quality of the controllers it produced, no experiments are reported which look at how repeatable the behavior of the individual controllers was.

ESs have also been found to be a viable, and possibly better alternative than GAs for evolutionary robotics. Salomon used an ES (with an added crossover operator) to evolve two different controllers for a Khepera involved in simple navigation and obstacle avoidance (Salomon, 1996, 1997).

Salomon chose to use an ES because they perform better at problems involving *epistasis*. Epistasis occurs when two or more fitness parameters interact in a nonlinear fashion, as is the case in most robotics applications. ESs also tend to converge more quickly on optimum solutions than GAs, and are therefore advantageous in situations using physical robots, where the time to obtain a (reasonable) solution is important.

Salomon used a similar experimental setup to that used by Mondada & Floreano (1995); Floreano & Mondada (1996b), so that he could compare his results using an ES with their results using a GA. Two neural network controllers were evolved:

1. A controller inspired by Braitenberg vehicle 3-c (Braitenberg, 1994), reported by Salomon (1996).

2. The evolution of receptive fields (Salomon, 1997). Receptive fields are more complex controllers requiring more parameters for the ES to optimize, (see Moody & Darken, 1988 for details).

Time required for training: Salomon found ESs to be an order of magnitude quicker at finding an equally good solution as the solutions produced by the GA used by Floreano and Mondada. This is a significant speed up of the training phase, showing that ESs have

a lot to offer evolutionary robotics, although very few researchers have used them. In Section 6.2 another project using an ES is reviewed.

Generality of the controller: There are no experiments reported that consider the generality of the evolved robot controllers.

Accuracy and repeatability: It was found that ESs worked well for both types of controller considered, showing that they can scale up from simple ones like the Braitenberg controller. However, as before, no experiments are reported which look at the issue of repeatability of the resulting controllers.

4.2 The "Simulated and Physical Robot → Physical Robot" Form of TPE (TPE:2)

This section examines methods of TPE which used both a physical robot and simulation in training, before porting to a physical robot. The aim of combining physical robots and simulation in training is to mitigate the problems inherent in evolving on a simulation alone, such as evolving solutions that do not map to the real world, and the problems of using physical robots, such as the time taken to train and the cost of possible damage to the robot due to highly unfit initial genotypes. Figure 3 shows where this section fits into the framework defined in Section 3.1.

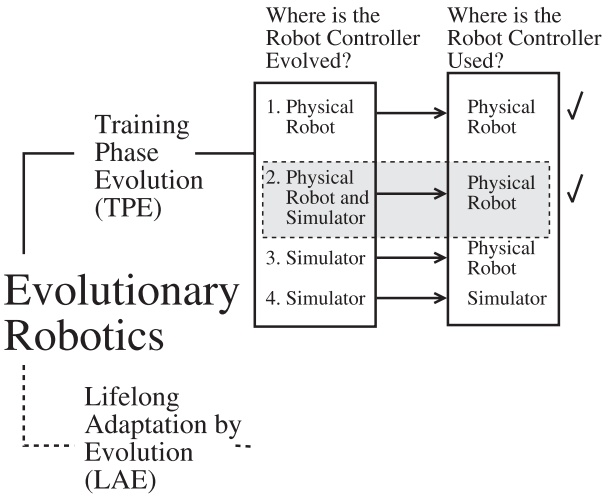

Figure 3 The position of Section 4.2 within the structure of this paper, as defined in Figure 1.

4.2.1 TPE:2—Simulation Followed by Fine Tuning on a Physical Robot

The work of Miglino and colleagues began by quantitatively analyzing the differences between the results of training in simulated and physical worlds (Miglino, Lund, & Nolfi, 1995a). As a result they developed a method that performed most of the evolutionary training phase in simulation, followed by a final fine-tuning of the behaviors on a physical robot (Miglino et al., 1995b). This approach was used to evolve a neural network controller for a Khepera robot that wandered its environment and avoided obstacles. The majority of evolution occurred in a simulation that was carefully built using sampled data from the physical robot's sensors. The GA was then run for 300 generations in the simulation, but when the results were transferred onto the physical robot, a drop in performance was recorded, so a further 30 generations were run in the real world.

Time required for training: The initial 300 generations took only an hour in simulation, but with 30 further generations being needed and each generation being made up of 100 individuals, this may have taken a significant amount of time, although the exact details are not reported. The time taken to sample the environment using the Khepera's sensors must also be taken into account, but again it is likely that this was time well spent, as a less accurate simulation would have led to more generations being required on the physical robot. However, it is likely to compare very favorably with the work of Floreano & Mondada (1996b) which took 10 days for the evolution on the physical robot.

Generality of the controller: No experiments are reported that explored the generality of the evolved controllers. The simulation was constructed to mimic a specific type of environment, therefore the robots evolved in it may not generalize well to different environments.

Accuracy and repeatability: The results in terms of the accuracy of performance of the final evolved neural networks were good, but again, no experiments were reported which looked at the repeatability of their performance.

4.2.2 TPE:2—Interleaving Simulation and Physical Robots

Wilson and others (Wilson, King, & Hunt, 1997; Wilson, 2000) introduce a methodology in

which evolution in simulation and the real world are interleaved. The evolutionary process was split into distinct phases with some phases in simulation and some on the physical robot. Firstly, primitive behaviors, such as *move-forward* and *turn-left*, were designed and tested on the physical robot, then these basic behaviors were randomly concatenated to create sequences of behaviors. These sequences were evaluated for fitness and could, at a later stage, be used as chunks in larger-scale sequences.

The robot's task was to travel a maze to a goal, and the evolutionary process combined simple behaviors into sequences. The fitness at each phase was based on the robot's ability to reach the goal in a maze. The first stage introduced sufficient variation into the population, using mutation as the main operator. The second stage reduced the number of individuals in the population and used crossover as the main operator. The third phase tested the fittest population members on the physical robot. The genotypes run on the robot were evaluated, and the best ones were *chunked* (i.e., a set of genotypes were treated as a single entity), so that they could later be incorporated in the population through the mutation operator. The process was then repeated, beginning from evolution with a high mutation rate.

Time required for training: The use of simulation for part of the training meant that training was faster than if real robots had been used throughout. However, the repeated re-testing of behaviors on the physical robot required frequent involvement by the experimenter. This compares with the approach of Miglino et al. which needed significant human input to create the simulation in the first place, but after that just a one-off phase of evolution at the end of the process, and the simulated world could certainly be used again for different tasks (if not different environments), whereas the only part of the training that would not need to be repeated in Wilson's approach would be the creation of basic behaviors.

Generality of the controller: Wilson et al. did not test the generality of their final controller in different environments to the physical training environment.

Accuracy and repeatability: The accuracy and reliability of the resulting behavior sequences was tested by running them many times in the same environment and recording how consistently they performed. It was found that although they reliably found the goal, they were not very accurately repeatable.

4.3 Summary of TPE Methods

This section has considered ten projects which have used TPE. Each of these had a different approach and differed in how well they fulfilled the criteria defined in Section 3.2.

Using physical robots for evolution is often more time consuming than using simulation, and for this reason only relatively simple worlds and/or tasks have been investigated when physical robots have been used. The use of alternative evolutionary algorithms to the usual GAs is promising in this respect, as shown by the work using ESs by Salomon (Salomon, 1996, 1997). It has also been shown with the work using GP by Nordin, Banzhaf, and others (for example Nordin & Banzhaf, 1995), that changes can be made to the method, such as not testing all genotypes, to significantly increase speed of development.

Using a mixture of simulation and physical robots is another promising approach, but the simulations must be accurate if a real robot is only used at the end of the training phase for fine-tuning, as by Miglino et al. (1995a). Interleaving simulated and real runs during the evolutionary process, as by Wilson et al. (1997), is a good way to address this issue, as the robot is being frequently tested and re-adapted to the real world, without having to spend too long in it; however, this requires heavy involvement by the trainer who must be almost continually present.

Most of the projects did not test their evolved controllers in new worlds, to assess their robustness in the face of new environments after training has ended; therefore no statement can be made about the usefulness of the evolved solutions outside of the niches in which they were evolved. One example where researchers did look at the issue of generalization after training was by Hornby et al. evolving quadruped gaits, who found that a more difficult training environment produced a more generalized solution (Hornby, Lipson, & Pollack, 2001; Hornby & Pollack, 2001).

All the authors whose projects have been reviewed here report that their methods produced controllers that could satisfactorily carry out their tasks, although Nolfi and Marocco's robot, with evolved vision, did not perform as well as a simulated robot in training

(Nolfi & Marocco, 2000). Only a few have specifically addressed this issue by comparing their results with hand-designed controllers, or to other evolutionary results. When reports of this nature have been made they are mostly favorable. In Salomon's work, using an ES, the results were compared to Floreano and Mondada's early work (Floreano & Mondada, 1994) and found to be as good, but produced much faster. In a comparison between gaits evolved for the Sony AIBO, Hornby et al. found the evolved gaits to be better than hand-designed ones. However, in their work evolving fuzzy controllers, Matellan et al. (1995, 1998) did not see an improvement over hand-design when using a GA.

5 A Survey of "Lifelong Adaptation by Evolution" (LAE)

As with the TPE section, the work in each part the LAE branch of Figure 1 has been grouped by research group or type of work. There are fewer examples of LAE than TPE, and most LAE has been done using some *combination* of physical and simulated robots. At the end of the section, the projects are discussed in the light of the evaluation criteria given in Section 3.2.2. The following groupings of methods are examined:

LAE in Physical Robots (LAE:1)
- Evolution embodied in a population of robots.
- Co-evolution.

LAE in Simulation and Physical Robots (LAE:2)
- "Anytime learning."
- "Anytime learning" for hexapod gaits.
- Evolving morphology and control.

5.1 The "Physical Robot ↔ Physical Robot Form of LAE (LAE:1)

The position of this section, within the framework defined in Section 3.1, is shown in Figure 4.

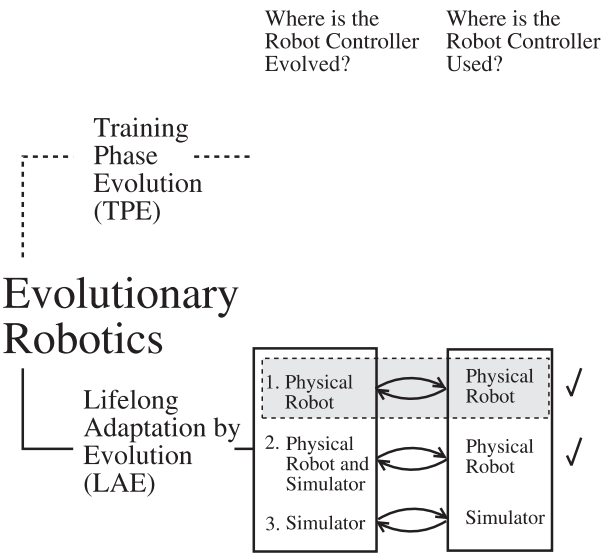

Figure 4 The position of Section 5.1 within the structure of this paper, as defined in Figure 1.

5.1.1 LAE:1—Evolution Embodied in a Population of Robots Using GAs, Watson et al. have explored how physical robots might continually adapt to a changing environment (Watson, et al., 1999; Watson, Ficici, & Polllack, 2000; Ficici et al., 1999). They name their approach "embodied evolution." A group of eight simple robots formed the GA's population, where each robot embodied a single genotype. The GA used was a version of Harvey's microbial GA (Harvey, 1996, unpublished work available at: ftp://ftp.cogs.susx.ac.uk/pub/users/inmanh/Microbe.ps.gz). The behavior of a robot was defined by its genotype, and each robot had a virtual energy level that indicated the fitness of its genotype. The task was to find a light source, and the virtual energy level (fitness) increased when the light was found.

Robots could mate when they met by broadcasting a mutated version of one of its genes, the rate of broadcast being proportional to its energy level, so a more fit robot was more likely to mate successfully. When a robot received a broadcast there was a probability, also based on its energy level, that it would overwrite its genotype with the mutated version of the other robot's genotype, so that a more fit robot was less likely to have its genotype overwritten.

The GA used required only minimal computation because the fitness function was simple; the amount of information transmitted was low (just a single mutated gene in each attempted mating), and the only evolutionary operator used was mutation. This means that, unlike many GA implementations, this one can practically be used on physical robots, and it is more likely to scale up to more complex tasks. It is especially appropriate for multi-agent tasks that naturally bring the robots into contact with each other for mating.

A similar project to that by Watson et al. has been reported by (Nehmzow, 2002). The major differences between the two projects are that there were only two robots in Nehmzow's experiments compared to eight in Watson et al.'s; Nehmzow's robots learned a larger number of behavioral competencies, and crossover rather than mutation was used. The robots were pre-programmed with basic behaviors such as obstacle avoidance. These pre-programmed behaviors were then improved by evolution, and new behaviors, such as phototaxis, were learned. As with Watson et al.'s work, the robots would attempt to mate when they met, after a period of testing their genotype. Each robot would transmit its genotype and that genotype's fitness, and the likelihood of crossover occurring between a robot's current genotype and the new one depended on the fitness of the two genotypes. In addition, each robot held a copy of the best genotype it had found so far which it would use if the GA did not produce a better genotype. It was found that this method of evolution optimized the behaviors quickly.

Watson's and Nehmzow's methods will be compared to the LAE criteria together as they are very similar approaches.

Adaptation in real time: Neither project tested the response of the evolutionary method to a dynamic environment. For both projects, results are presented that show steady increases in performance from the first generation until the behavior competence had been achieved, but there is no indication of the actual amount of time it took to adapt the robot controllers to their environments. However, Nehmzow found that his system could adapt to new tasks successfully, although the time taken to adapt is not presented.

Overall improvement in performance: The speed with which evolution can progress in this type of method is determined by how often the robots come into contact with one another. For tasks and environments where robots will frequently come into mating range with each other, the speed of evolution will be faster than when robots are not in often in close proximity. Although results are presented that show steady increases in performance over time for the two projects, there is no indication of the actual amount of time it took to adapt the robot controllers to their environments, nor is there any indication of how much the robots continue to adapt over the long term.

Interference of the evolutionary process in the robot's task: As long as the robots will naturally come into contact with one another during the progress of their task, and because the amount of genetic material to be transferred during each "mating" was small (especially in Watson's work), the method used in these two projects interfered very little in the task of the robots.

5.1.2 LAE:1—Co-evolution Co-evolution is the evolution of two or more agent behaviors which interact with each other, usually competitively, so that changes in the behavior of one agent drives further adaptation in the other, and evolution continues in an open-ended fashion.

In robotics the most widely studied case concerns a prey robot and a predator robot, for example Koza (1991) and Reynolds (1994). The predator tries to capture the prey and uses a GEC to find better ways to catch the prey. At the same time, the prey adapts by GEC, to avoid the predator. As each agent finds new ways to achieve their goal, so each must evolve new ways to frustrate the other. A similar idea is the evolution of strategies for playing robot soccer, where robots must learn to work as a team and compete with an opposing team to score, for instance Ostergaard and Lund (2003). Co-evolution is an inherently lifelong process, but has rarely been implemented on real agents. One example where real robots have been used is an on-going project by Floreano and others (Floreano et al., 2001).

In their work Floreano et al. used two Khepera robots, one as the predator and one as the prey. The predator had to catch the prey by touching it. A workstation was used to carry out all the GA computation, and each robot received the next genotype after the end of each trial. Trials ended when the predator caught the prey.

In the first few generations the predator scored low and the prey high, because the predators were not very good at catching the prey; however it took just 20 generations for the two agents to become more evenly matched. The authors suggest that competitive co-evolution can be used to overcome the "bootstrap" problem: i.e., when dealing with a complex task it is unlikely that an individual in the first, randomly generated generation will have even part of a solution to the task, so the whole generation receives a low fitness, meaning there is little selection pressure. Incre-

mental approaches can solve this problem but require varying amounts of human intervention. Competing robots, however, have been shown to produce increasingly difficult challenges for each other, without any outside help (Nolfi & Floreano, 1998). An indefinitely increasing performance is not guaranteed because the robots tend to cycle through the same solutions.

Adaptation in real time: Both robots were able to adapt quickly to the behavior of the other robot, however the actual amount of time taken for adaptation to occur is not reported (just the fact that it took 20 generations).

Overall improvement in performance: An overall improvement in performance was seen for the first 20 generations, but after that oscillations in performance set in, with first one, then the other robot performing better, and neither improving its performance overall.

Interference of the evolutionary process in the robot's task: The evolutionary process took place on board a computer to which the robots were tethered throughout experimentation; this would not be practical for most real world applications. However, the tethering was due to the small amount of memory and processor power on the robots themselves and could be avoided, either by using radio links or larger robots.

5.2 The "Simulated and Physical Robot ↔ Physical Robot" Form of LAE (LAE:2)

The position of this section, within the framework defined in Section 3.1, is shown in Figure 5.

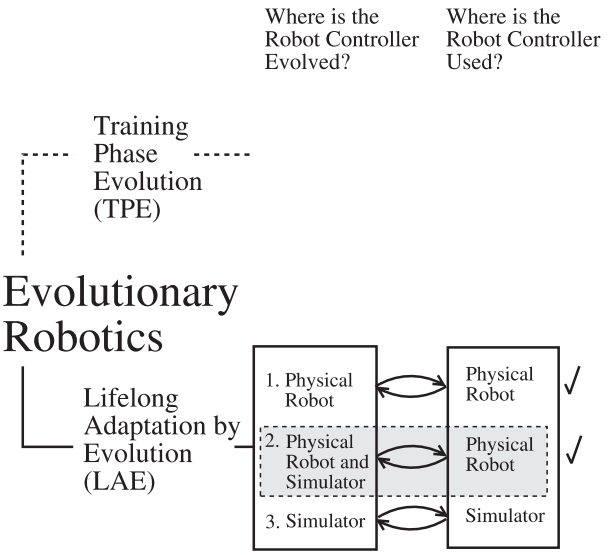

Figure 5 The position of Section 5.2 within the structure of this paper, as defined in Figure 1.

5.2.1 LAE:2—"Anytime Learning" The foundations of *anytime learning* is SAMUEL, introduced by Grefenstette, Ramsey and Schultz (Grefenstette et al., 1990). SAMUEL is a learning system that learns reactive behaviors defined by stimulus–response rules; it has two parts. Firstly, a learning system that uses a GA running in an off-board simulation to continuously adapt the robot to its environment and task. Secondly, an execution system that uses the rules provided by the learning system to control a robot. The system has been implemented on physical robots successfully on tasks such as collision avoidance (Grefenstette & Schultz, 1994) and missile evasion (Schultz & Grefenstette, 1992). However, SAMUEL's simulator was not updated with information from the robot's interaction with the world, making it much harder for it to adapt to new situations.

Another version of SAMUEL has been designed to perform anytime learning (Grefenstette & Ramsey, 1992), which is an algorithm that receives information from the real world to allow evolutionary adaptation to update the simulator. A *monitor* was added to the execution system, which notified the learning system whenever the world had changed, and the simulation environment would then be changed accordingly. The ability to change the simulation in which learning takes place means that the robot can adapt to new environments that were not initially designed into the learning system, making the SAMUEL architecture much more robust.

SAMUEL has been tested using a "cat and mouse" scenario: the cat robot must learn to keep the target mouse robot in range without being detected; the mouse robot moves randomly until it detects the cat robot, when it runs away at high speed. Experiments were carried out to discover how quickly the system could adapt to a changing environment, by altering the speed of the mouse robot. Whenever the monitor detected a change in the speed of the mouse it updated the simulation with the new information, and whenever the simulation found a better strategy than the one currently being executed it passed that to the

robot. A comparison was made between a robot with and without this form of adaptation. The results showed anytime learning had a distinct advantage.

A further extension of SAMUEL involved the addition of case-based reasoning (Ramsey & Grefenstette, 1993). Good genotypes were saved, and stored with description of the environment they performed well in. When the environment changed, the GA was restarted, and the new population seeded with saved genotypes from similar environments to the new one, as well as with default strategies that were known to perform well in a range of situations. The method was successful, but the robot had to be able to recognize when the environment had changed, and be able to compare that environment to previous ones, a difficult problem for most robots.

Adaptation in real time: The robot was able to adapt to the changing speed of the mouse, and continued to be able to catch it during experimentation. The actual amount of time it took to adapt is not reported.

Overall improvement in performance: Tests showed that sometimes when the environment changed to become more difficult (when the mouse sped up significantly), the robot did not adapt to reach previous levels of performance before the next change in the environment. However, the robot may not be able to perform as well when the mouse moved more quickly, however much it adapts. At other times the fitness for a given environment (or mouse speed) exceeded previous levels, indicating that the system was producing an overall improvement.

Interference of the evolutionary process in the robot's task: The robot and the learning system communicated with each other at intervals which meant that the robot's task would be interrupted. This interruption could be significant if the robot was tethered, or needed to move to be within communication range of the workstation, but otherwise would probably not cause significant problems for task execution. It is worth noting, however, that as an environment changed more frequently, more communication would be required, both from the robot informing the monitor of change, and from the learning system updating the robot controller.

5.2.2 LAE:2—"Anytime Learning" for Hexapod Gaits Parker and Rawlins devised "cyclic GAs" for the evolution of gaits (Parker, 1998; Parker & Rawlins, 1996). These encode genotypes as a circle with two tails, rather like a "$\propto$" symbol, where the tails can alter in length. This replaced the standard linear chromosome. These partially cyclical GAs were so designed because the execution of a gait consists of cycling through a set of actions, with the two tails encoding the initial and final action sequences.

In order for the robot to continually adapt to a changing environment Parker altered a version of Grefenstette et al.'s anytime learning, which they called "punctuated anytime-learning" (Parker & Mills, 1999; Parker, 2000b). Evolution took place in simulation, with the best solution sent to the physical robot periodically, and information from the robot being used to alter the learning process by initiating changes to the GA (rather than the simulation as in SAMUEL).

After a number of generations, all the genotypes in the current population were tested on the physical robot and, if there were discrepancies between the fitness a genotype gained in the simulation and the fitness it gained in the real world, a fitness "bias" was calculated for it. This bias was given by a genotype's fitness on the real robot divided by its fitness on the simulated robot. In subsequent generations, a genotype's bias is the average of its parents' biases. A genotype's fitness is given by multiplying its initial fitness by its bias.

Initial work using this system was performed on simulated robots but more recent results from testing the method on a physical robot (Parker, 2000a; Parker. & LaRochelle, 2000) were encouraging. However, the evolutionary process had to be simplified (and weakened) because it interrupted the robot in achieving the task. This was done by reducing the GA's population size and number of times the genotypes were evaluated, and by cutting down the number of times the simulation performance was tested.

Subsequent work using punctuated anytime learning has looked at: Incrementally evolving individual leg controllers (Parker, 2003a); adapting to a different environments (Parker, 2003b) and the evolution of a simulated team of legged robots (Parker & Blumenthal, 2002).

Adaptation in real time: Punctuated anytime learning has been used to adapt gaits to changes in an environment (Parker, 2003a); however, no clear indication of the amount of time it took to readapt after the environment changed was reported—only that after 200 generations performance was still improving, but had not reached previous levels.

Overall improvement in performance: An improvement in performance was seen after environmental change; however, there are no long term experiments reported which look at repeated changes to the environment and their effect on a physical robot's performance.

Interference of the evolutionary process in the robot's task: The amount of interference with a robot's task is similar to that for Grefenstette's Anytime Learning. Using radio for transmission reduces the amount of interference, but the more changeable an environment, the more interruptions will be needed.

5.2.3 LAE:2—Evolving Morphology and Control

Field programmable gate arrays (FPGAs) are reconfigurable electronic circuits made up of logic gates that can be configured by GEC. The first use of FPGAs and evolution to control a robot was made by Thompson in 1995 (Thompson, 1995), who evolved wall-avoidance behavior in a wheeled robot.

Keymeulen et al. have more recently evolved an FPGA to control a visually guided robot (Keymeulen, Iwata, Kuniyoshi, & Higuchi, 1998a, b). The robot's task was to follow a ball whilst avoiding obstacles, and it was expected to adapt to new situations such as movement of obstacles and loss of individual sensors. Two FPGAs were used—one executed the current best genotype and one used a simulation of the world to perform evolution, so that although evolution took place in a simulation, the simulation was run on the robot itself. The model of the world used in the simulation was built by the robot as it moved around, and after every ten generations the best genotype to date was transferred to the first FPGA for execution.

Adaptation in real time: It was found that this method could evolve fit controllers in just 5 min, and that it could thus adapt to new situations very quickly.

Overall improvement in performance: The system was able to adapt to environmental changes, and showed an overall increase in performance over time.

Interference of the evolutionary process in the robot's task: Using a simulator for evolution, and situating that simulator onboard the robot itself allowed the robot to refer to simulation without interrupting its task significantly, as no medium or long distance communication was needed with a computer workstation.

5.3 Summary of LAE Methods

This section has considered five projects using LAE, either on physical robots alone, or on a mixture of simulated and physical robots.

For real-time evolution to be useful, it needs to be able to adapt quickly to changing environments, since different environments will present different challenges. Some researchers have failed to investigate this issue; however, Nehmzow's work with a team of robots showed how an embodied GEC can learn new tasks (Nehmzow, 2002), Ramsey and Grefenstette showed how SAMUEL adapted to changes in one environmental variable (Grefenstette & Ramsey, 1992), and the work of Floreano et al. showed reasonably high-speed reaction to new conditions (Floreano, Nolfi, & Mondada, 2001).

In all the examples looked at, evolution produced an overall, long-term improvement in performance, although comparisons are rarely made between robots using LAE methods, and robots that did not use them. The work with SAMUEL is one exception to this, and it was shown that the anytime learning method did provide improvements in performance.

Many of the approaches discussed have required some level of interference by the chosen GEC in the task. In Parker et al.'s work using punctuated anytime learning for hexapod gaits (Parker & Mills, 1999; Parker, 2000a), the robot was required to be in contact with the workstation performing the computation, if only occasionally. The work by Watson and others (Watson et al., 1999, 2000; Ficici et al., 1999) and Nehmzow has addressed this issue in part by embodying evolution in a team of robots, making evolution a multiagent task. Alternatively, using a simulator for evolution, and situating that simulator onboard the robot itself, as by Keymeulen et al. (1998a, b), allowed a

simulation to be used without interrupting the robot's task unduly.

6 Combined TPE and LAE Approaches

In this section, two projects are reviewed that have combined TPE and lifelong adaptation. The first project only uses evolution for adaptation during training, and then uses non-evolutionary, "plastic neural networks" to continue adaptation after training. The second project brings together the both the TPE and LAE branches of Figure 1 in a way that uses evolution in both sections of its method. The criteria are applied to both projects together in Section 6.3.

6.1 The Approach of Uzrelai and Floreano

In order to give evolved robots robustness in the face of new environments, Uzrelai and Floreano used a GA to evolve plastic neural networks (PNNs) during a training phase (Floreano & Urzelai, 2000a). After training, the PNN allowed adaptation to continue over the lifetime of the robot. The genotypes encoded four modification rules (plain Hebb, post-synaptic, pre-synaptic, and covariance) for the PNN. Each rule could be altered genetically. When a genotype was tested on a robot the synaptic strengths of the neural network were set to random values, and synaptic strengths of the PNN were adapted over the course of the run according to the modification rules encoded in the genotype.

Introduced by Floreano & Mondada (1996a), the method was first used to learn navigation and obstacle avoidance. In Floreano & Urzelai (2001) the learning rate of this approach was compared to that of a normal neural network plus a GA, and they found that it took fewer generations, of smaller populations, to evolve good solutions. In Urzelai & Floreano (2000a) the same was found for the more complicated light-switching task. This involves placing a robot in a rectangular environment with a light at one end and a switch for the light at the other end and the robot has to switch the light on and then move underneath it to gain fitness.

A series of experiments tested how well the evolved robots could adapt to new environments after the training phase:

- Moving from simulation to reality. Robots were evolved in simulation, then transferred to reality (Urzelai & Floreano, 2000b).
- Changing spatial relationships. The position of the light was changed in the light switching task (Urzelai & Floreano, 2000b).
- Changing the robot platform. A controller evolved on a Khepera was transferred to a larger robot with different sensor capabilities (the Koala robot), again for the light switching task (Floreano & Urzelai, 2000b).

In all cases it was found that the evolved controllers transferred to the different environments quickly and successfully.

6.2 The Approach of Walker et al.

Work by Walker (Walker, 2003; Walker & Wilson, 2002) investigated the relative contributions of TPE and LAE to robot performance. A standard GA was used for TPE, and a bespoke, minimal ES was used for LAE. The training phase was intended to produce a robust robot controller that would be general to many situations, including several new ones, without further adaptation. The LAE provided the ability to continue to improve performance, and the ability to evolve solutions to situations beyond the generality of the controller produced by TPE.

TPE took place over 500 generations on a simulated Khepera robot. The robot's task was to find a goal quickly, whilst avoiding obstacles. A GA was used to optimize a set of Arkin's behavioral schemas (Arkin, 1989) and, throughout training, the world changed both in terms of the locations of obstacles, and the obstacle density. This continuously dynamic world produced a robot controller that was shown to generalize to new worlds after training, and in fact often performed better in these worlds than controllers specifically evolved for them and them alone.

LAE was then used to adapt further the robot controller, using a novel version of an ES. The task of the robot was similar to its task during training: it had to move towards the light source goal whilst avoiding obstacles. However, once a goal had been found, the light would be switched off, and another switched on to provide the next goal. Experiments were carried out in a number of different dynamic environments, in which the number of obstacles changed at different rates and with different frequency.

The ES was designed to be minimal, to fit on almost any robot, having a very small population size

of three chromosomes, and used only the mutation operator. A novel technique was used to offset the instability caused by having such a small population size in a dynamic world: rather than the best genotype automatically replacing the parent in each generation, it was saved and only replaced the parent if it also performed well in the next generation (Walker, 2003).

Experiments, reported by Walker (2003), tested the algorithm in simulation, and showed that the evolved controller ported successfully to a physical Khepera. Further experiments were carried out on a physical Khepera robot. The conclusions that have been drawn from this work are:

- Even a minimal LAE algorithm can improve the performance of a trained robot controller in dynamic environments.
- The improvement seen is compromised when the environment changed suddenly, massively, and/or frequently.
- A training phase is vital to the performance of the minimal LAE algorithm.

This work is apparently unique in combining TPE and LAE, both using evolution for adaptation, and in testing the algorithm in a variety of dynamic worlds to assess its robustness and generalizability.

6.3 Applying the Evaluation Criteria to the Combined Approaches

As these approaches use both TPE and LAE, both sets of criteria are relevant for their analysis. However, the focus of Urzelai and Floreano's work was on the lifelong adaptation, and so reports are not available of the trained robot's robustness and accuracy before that phase began.

TPE Analysis:

Time required for training: In Walker's work the use of a simulation allowed the robot to be quickly and easily tested in a continuously changing world, something that would have been much harder to achieve in a physical training environment.

Generality of the controller: Walker's training phase was designed to produce a robot that was immediately robust to new situations, and tests were carried out show that the evolved controller could indeed transfer from simulation to reality, and that it could perform well in a range of environments with significantly different obstacle densities.

Accuracy and repeatability: In Walker's work, the evolved controller was found to transfer to the physical robot well, although the transfer was not entirely accurate—the physical robot performed well with good repeatability, even more so than the simulated robot.

LAE Analysis:

Adaptation in real time: Both Urzelai and Floreano's, and Walker's work showed quick adaptation to changes in the environment. Walker tested the limits of the ES-based LAE algorithm in dynamic environments. Floreano et al's work in evolving plastic neural networks showed that it was possible to produce successful adaptation to a variety of new environments and situations (Floreano & Mondada, 1996a) using their method.

Overall improvement in performance: Both projects showed an overall improvement in performance for the robot when using lifelong adaptations. Walker compared a robot using the LAE algorithm with one without and found a significant improvement in performance except in very rapidly changing environments.

Interference of the evolutionary process in the robot's task: As lifelong adaptations occurred onboard the robots themselves, neither of the projects reviewed in this section involved robot's interrupting their task to communicate with a workstation or other robot, however both were tethered for power and communication and/or results capture purposes and may have had their motion impeded to some extent.

7 Discussion

7.1 New Directions

In both TPE and LAE there are various possible combinations of simulation-based and physical robot-based evolution; some of these possibilities have been discussed above, but some have not. These possibilities are tabulated in Tables 1 and 2. The rows of these tables are labelled with the location of the evolution-

Table 1 Feasible and infeasible approaches for TPE; known approaches are discussed in the main text. (✓ indicates a feasible approach, × indicates an infeasible approach, () indicate a degree of doubt)

Evolution/agent type	Simulation	Simulation and real robot	Real robot
Simulation	✓	×	✓
Simulation and real robot	✓	(✓)	✓
Real robot	(✓)	×	✓

ary method (i.e., does evolution occur in a simulation or on a physical robot, or to some combination?) and the columns refer to the type of agent to which the resulting, evolved robot controller is applied (i.e., to a simulated robot, a physical robot, or to some iterative combination?).

7.1.1 Possibilities in TPE Methods Table 1 contains nine possible approaches to TPE. Consider the three possible applications of TPE that occur solely in a simulated world. The work of Sims (1994) and Mautner and Belew (1999) shows that it can be applied to a *simulated robot*, and the work of Jakobi, Husbands, and Harvey (Jakobi et al., 1995) shows that it can be applied to a *physical* robot. However, it does not appear to be particularly useful to apply a controller developed in simulation to both robot and simulation, except perhaps to compare the results from two application domains. Note that this information from the physical world cannot be fed back to be used in further training, because the method is restricted to adaptation from simulated data alone.

Next consider a training phase where a controller is evolved using information drawn from both simulated and physical worlds. This may involve two separate populations of controllers being evolved, one from simulated data and one from physical data, or there may be a single population that is tested in both simulated and physical worlds. In both cases, if the evolved controller is then used on a simulated robot, it may be more robust than it would otherwise have been, since it will have evolved to cope with noise from the real world; if the controller is then used on a physical robot, it may have been evolved more quickly than it otherwise would have been (Wilson, 2000; Nolfi et al., 1994). It is also conceivable that the evolved controller may be ported to both a simulated and a physical robot, since this would again allow for a comparison of results. In addition, perhaps a candidate controller that performs well in both domains should be seen as more fit than one that performs well in only one domain. Nevertheless, the simulation of the world itself may still limit the realism of any results in this case.

Finally, a controller evolved on a physical robot alone can obviously be applied to a physical robot (Floreano & Mondada, 1996b), but it might also be ported to a simulation if it was important to do fast, off-line experimentation on its task, although the usefulness is somewhat in question because the accuracy of the results (relative to the real world) would then depend on the accuracy of the simulation. It is hard to imagine a situation where it would be useful to port the results to both types of world, other than for experimental comparison of results, because information from the simulated robot cannot be used in further training.

7.1.2 Possibilities in LAE Methods Table 2 contains the nine possible approaches to LAE. Consider the case where the robot controller is developed by LAE in simulation alone. Clearly such a controller can be usefully used in a simulated world, where it is a lifelong extension of the corresponding training phase case. However, its application to a physical robot (or to the simulated and physical robots) is meaningless in the lifelong case, since LAE would require feedback from the physical robot and this row of the table concerns the type of LAE that occurs only in simulation.

There is a general principle here: in all three table rows, the type of agent on which evolution occurs must be the same as the type of agent it is applied to because the LAE learning loop cannot otherwise be properly closed. This was not true with training phase methods, where there was no feedback after training was complete, so it was possible to train on one type

Table 2 Feasible and infeasible approaches for LAE; known approaches are discussed in the main text; (✓ indicates a feasible approach, × indicates an infeasible approach)

Evolution/agent type	Simulation	Simulation and real robot	Real robot
Simulation	✓	×	×
Simulation and real robot	×	✓	×
Real robot	×	×	✓

of world and then to apply it to another type of world. Therefore, the three meaningful implementations of LEA require the feedback between the evolution and application of the robot controller to be related to the same type of world.

The second row of the table requires additional comment. Here evolution is taking place both in simulation and on the physical robot and feedback is obtained from both types of world. As discussed above, this might be used to help constrain the search for fitter genotypes, but now there is an extra overhead associated with having two world models active simultaneously: the combined cost of running both forms of evolution may make this option seem less attractive that the other two. Nevertheless, we have seen that periodically updating the simulation with information about the real world from the robot has been used successfully by Grefenstette & Ramsey (1992) and Parker (2000a).

This outlines all the possible combinations of location for evolution and application, for both training phase and LAE. But there are still other combinations! Any of the useful training phase methods described above may precede any of the useful LAE methods, as long as the evolved controller can be transferred to the chosen domain for the LAE methods: as mentioned, the LAE methods require feedback between the identical domain; for example, the controller from a simulated robot would need to ported to a physical robot if physical robot LAE were to be used, as by Walker (2003). Since this gives 21 combinations—many of them fairly pointless—the following section attempts to reduce them to the interesting possibilities.

7.1.3 Interesting, Novel Combinations of TPE and LAE Methods It seems sensible that in most settings the starting place for evolution would be a simulation. This is because to do otherwise risks damaging a robot as its initial random controller genotype may send it dashing into harm, and beginning with hand-designed controllers defeats the object of learning. Just considering simulated, or simulated-plus-real TPE followed by the three possible types of LAE reduces the possible combinations of TPE and LAE to six.

If simulated TPE were to be followed by simulated LAE there would at no point be application of the controller to a real robot; it has already been explained why such systems are not being explored here. Performing TPE in simulation and LAE exclusively on a physical robot has already been discussed (for example, Walker, 2003).

Applying simulation-only TPE to a simulated *and* physical robot LAE session would allow evolution to be made progressively more realistic by input from the real world, while minimizing the initial risk of damage to the real robot. This might even be implemented as a physical robot that could *simulate* intended actions before taking them to check they would not be dangerous, perhaps preventing wrong actions before they are actually made in the real world. No examples have been found of combining a training phase with this type of LAE.

The remaining three possibilities follow simulation-plus-real-robot TPE with the three possible types of LAE. Each of these approaches may involve more risk to the robot than simulation-only TPE, but they might have the advantage that real data are being used in adaptation for more (or all) points of the robot controller's evolution. Further work in this area would be of value.

8 Conclusions

In this paper we have seen several approaches to adaptation by evolution in robotics. Evolution may be used

exclusively in a training phase prior to a task, or lifelong during a task or set of tasks, or it may involve a combination of the two. In all cases, however, the aim is to make the training of a robot more efficient than it would be by hand and this is not always achieved. A number of conclusions can be drawn:

- Initial evolution on a physical robot is problematic at best and damaging to the robot, or its environment, at worst.
- Evolution in simulation usually requires some input from a physical robot to produce useful results, although a good simulation can minimize these problems.
- TPE may not be able to adapt to all that the robot might experience, limiting the performance of the robot.
- LAE requires that the learning loop is closed and therefore restricts the results of evolution to the same type of agent as that on which the evolution is executed; all three possible choices of agent have been seen to have their limitations.
- The benefits of combined TPE and LAE can be obtained by preceding the LAE on a physical robot with a TPE phase that includes an element of simulation; other types of training phase appear to be less useful, although there are a number of combinations still to be explored.

A combination of simulated and real robots appears to offer real benefits in the evolution of robot controllers. This is not perhaps surprising since psychologists believe that humans also perform both types of activity when learning a task (Bruner, 1964). Drawing further on these insights may stimulate the development of still better learning methods. For example, the ability to interpolate and extrapolate in simulated behavior appears to be useful in human thinking: we can remember physically jumping off a wall and extrapolate, in simulation, to what might happen if we jumped off a cliff, and recoil from the results.

The work presented here has suggested that the use of simulation has a place in evolutionary research on physical robots, and made some suggestions for where is best to use it. Improvements are still required in terms of the speed and accuracy of adaptation, and in this respect adaptation by evolution may need to be combined with other methods of adaptation; hopefully, some of the strategies suggested here for combining TPE and LAE, as well as the possibility of using simulation data alongside data from a real robot, may help to open up new avenues of research in this area.

Note

1 For the remainder of the paper we use the term robot to mean a mobile robot; we do not consider robot arms, or other types of robot.

References

Arkin, R. C. (1989). Motor schema–based mobile robot navigation. *International Journal of Robotics Research*, *8(4)* 9–12.

Back, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, *1(1)* 1–23.

Bajcsy, R. (1988). Active perception. *Proceedings of the IEEE*, *76(8)* 996–1005.

Banzhaf, W., Nordin, P., & Olmer, M. (1997). Generating adaptive behavior using function regression within genetic programming and a real robot. In *2nd International Conference on Genetic Programming*, (pp. 35–43) San Francisco: Morgan Kaufmann.

Bongard, J. (2002). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, (pp. 1872–1877). IEEE Press.

Booker, L., Goldberg, D., & Holland, J. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, *40*, 235–282.

Braitenberg, V. (1994). *Vehicles: Experiments in Synthetic Psychology.* MIT Press, Cambridge, Massachusetts.

Brooks, R. (1986). Achieving artificial intelligence through building robots. A.I. Memo 899, Massachusetts Institute of Technology Artificial Intelligence Lab.

Brooks, R. (1992). Artificial life and real robots. In *European Conference on Artificial Life*, (pp. 3–10).

Bruner, J. S. (1964). The course of cognititve growth. *American Psychologist*, *19*, 1–15.

Colombetti, M. & Dorigo, M. (1992). Learning to control an autonomous robot by distributed genetic algorithms. In *From Animals to Animats 2, Proceedings of the 2nd International Conference on the Simulation of Adaptive Behavior SAB-92*, (pp. 305–312).

Dorigo, M. (1995). ALECSYS and the AutonoMouse: Learning to control a real robot by distributed classifier systems. *Machine Learning*, *19(3)* 209–240.

Ficici, S., Watson, R., & Pollack, J. (1999). Embodied evolution: a response to challenges in evolutionary robotics. In

8th European Workshop on Learning Robots, (pp. 14–22).

Floreano, D. & Mondada, F. (1994). Automatic creation of autonomous agent: Genetic evolution of a neural-network driven robot. In *Proceedings of 3rd International Conference on Simulation of Adaptive Behavior, (SAB-94)*, (pp. 421–430).

Floreano, D. & Mondada, F. (1996a). Evolution of plastic neurocontrollers for situated agents. In *From Animals to Animats 4, Proceedings of the 4th International Conference on the Simulation of Adaptive Behavior (SAB-96)*.

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

Floreano, D., Nolfi, S., & Mondada, F. (1998). Competitive co-evolutionary robotics: from theory to practice. In *From Animals to Animats 5, Proceedings of the 5th International Conference on the Simulation of Adaptive Behavior SAB'98*, Cambridge, MA: MIT Press.

Floreano, D., Nolfi, S., & Mondada, F. (2001). Co-evolution and ontogenetic change in competing robots. In M., Patel, V., Hanover, and K., Balakrishnan (Eds.) *Advances in the Evolutionary Synthesis of Intelligent Agents*, Cambridge, MA: MIT Press.

Floreano, D. & Urzelai, J. (2000a). Evolutionary on-line self-organisation of autonomous robots. In *Proceedings of the 5th International Conference on Artificial Life and Robotics*.

Floreano, D. & Urzelai, J. (2000b). Evolutionary robots with on-line self-organisation and behavioral fitness. *Neural Networks*, *13,* 431–443.

Floreano, D. & Urzelai, J. (2001). Evolution of plastic control networks. *Autonomous Robots*, *11*, 311–317.

Fogel, L., Owens, A., & Walsh, M. (1966). *Artificial Intelligence through Simulated Evolution*. New York, Wiely.

Gallagher, J., Beer, R., Espenschied, K., & Quinn, R. (1994). Application of evolved locomotion controllers to a hexapod robot. *Robotics and Autonomous Systems*, *19*, 95–103.

Goldberg, D. E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*. Reading, MA: Addison Wesley.

Grefenstette, J. & Ramsey, C. (1992). An approach to anytime learning. In *Proceedings of the 9th International Machine Learning Conference* (pp. 189–195) Amsterdam: Elsevier.

Grefenstette, J., Ramsey, C., & Schultz, A. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, *5(4)* 35–81.

Grefenstette, J. & Schultz, A. (1994). An evolutionary approach to learning in robots. In *Machine Learning Workshop on robot learning*, New Brunswick, NJ.

Harvey, I. (1997). Artificial evolution and real robots. *Artificial Life and Robotics*, *1,* 35–38.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.

Holland, J. (1993). *Adaptation in Natural and Artificial Systems, an Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA: MIT Press.

Hornby, G., Fujita, M., Takamura, S., Yamamoto, T., & Hanagata, O. (1999). Autonomous evolution of gaits with the Sony quadruped robot. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO'99)*, (pp. 129–304).

Hornby, G., Lipson, H., & Pollack, J. (2001). Evolution of generative design systems for modular physical robots. In *IEEE International Conference on Robotics and Automation*. Piscataway, NJ: IEEE Press.

Hornby, G. & Pollack, J. (2001). Evolving {L}-systems to generate virtual creatures. *Computers and Graphics*, *25(6)* 1041–1048.

Hornby, G., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., & Fujita, M. (2000). Evolving robust gaits with AIBO. In *IEEE International Conference on Robotics and Automation*, (pp. 3040–3045).

Jakobi, N. (1994). Evolving sensorimotor control architectures in simulation for a real robot. Master's thesis, University of Sussex.

Jakobi, N., Husbands, P., & Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life: Proceedings of the 3rd International Conference on Artificial Life*, (pp. 704–720) Berlin: Springer-Verlag.

Kato, T. & Floreano, D. (2001). An evolutionary active-vision system. In *Proceedings of the Congress on Evolutionary Computation (CEC'01)* (pp. 107–114). Piscataway, NJ: IEEE Press.

Keymeulen, D., Iwata, M., Kuniyoshi, Y., & Higuchi, T. (1998a). Comparison between an off-line model-free and an on-line model-based evolution applied to a robotics navigation system using evolvable hardware. In *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, (pp. 199–209).

Keymeulen, D., Iwata, M., Kuniyoshi, Y., & Higuchi, T. (1998b). Online evolution for a self-adapting robotic naviagation system using evolvable hardware. *Artificial Life*, *4,* 359–393.

Koza, J. (1991). Evolution and co-evolution of computer programs to control independently acting agents. In *From Animals to Animats. Proceedings of the 1st International Conference on Simulation of Adaptive Behavior, (SAB'91)*, (pp. 366–375). Cambridge, MA: MIT Press.

Koza, J. (1992). *Genetic Programming*. Cambridge, MA, MIT Press.

Lewis, M., Fagg, A., & Solidum, A. (1992). Genetic programming approach to the construction of a neural network for control of a walking robot. In *IEEE International Conference on Robotics and Automation*, (pp. 2618–2623). Piscataway, NJ: IEEE Press.

Marocco, D. & Floreano, D. (2002). Active vision and feature selection in evolutionary behavioural systems. In *From Animals to Animats: Proceedings of the seventh international conference on the simulation of adaptive behviour (SAB'02)*, (pp. 247–255).

Mataric, M. & Cliff, D. (1996). Challenges for evolving controllers for physical robots. *Robotics and Autonomous Systems*, *19(1)* 67–83.

Matellan, V., Fernandez, C., & Molina, J. (1998). Genetic learning of fuzzy reactive controllers. *Robotics and Autonomous Systems*, *25*, 33–41.

Matellan, V., Molina, J., Sanz, J., & Fernandez, C. (1995). Learning fuzzy reactive behaviours in autonomous robots. In *4th European Workshop on Learning Robots* (pp. 45–56). Karlsruhe, Germany.

Mautner, C. & Belew, R. (1999). Evolving robot morphology and control. In *Proceeding of the Artificial Life and Robotics Conference (AROB-99)*.

Miglino, O., Lund, H., & Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. *Artificial Life*, *2*, 417–434.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press.

Mondada, F. & Floreano, D. (1995). Evolution of neural control structures: some experiments on mobile robots. *Robotics and Autonomous Systems*, *10*, 221–249.

Mondada, F., Franzi, E., & Ienne, P. (1993). Mobile robot miniaturisation: a tool for investigation in control algorithms. In *Experimental Robotics 3: Proceedings of the 3rd International Symposium on Experimental Robotics* (pp. 501–513). Berlin: Springer-Verlag.

Moody, J. & Darken, C. (1988). Learning with localised receptive fields. In *Proceedings of the 1988 Connectionist Models Summer School*, (pp. 13–43). San Francisco: Morgan Kaufmann.

Nehmzow, U. (2002). Physically embedded genetic algorithm learning in multi-robot scenarios: The PEGA algorithm. In *2nd International Workshop on Epigenetic Robotics: Modelling Cognitive Development in Robotic Systems*.

Nolfi, S. & Floreano, F. (1998). How co-evolution can enhance the adaptive power of aritificial evolution: implications for evolutionary robotics. In *Proceedings of EvoRobot'98*, (pp. 2–8). Berlin: Springer-Verlag.

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

Nolfi, S. and Marocco, D. (2000). Evolving visually-guided robots able to discriminate between different landmarks. In *From Animals to Animats 6. Proceedings of the sixth International Conference on Simulation of Adaptive Behavior (SAB'00)*, (pp. 413–419). Cambridge, MA: MIT Press.

Nolfi, S., Miglino, O., & Parisi, D. (1994). Phenotypic plasticity in evolving neural networks. In *Proceedings of the International Conference from Perception to Action*, (pp. 14–57). Los Alamitos, CA: IEEE Computer Society Press.

Nordin, P. & Banzhaf, W. (1995). Genetic programming controlling a miniature robot. In *Working notes of the AAAI-95 Fall Symposium Series, Symposium on Genetic Programming*, (pp. 61–67). Cambridge, MA: AAAI.

Nordin, P. & Banzhaf, W. (1997). Real time control of a Khepera robot using genetic programming. *Cybernetics and Control*, *26(3)* 53–61.

Nordin, P., Banzhaf, W., & Brameier, M. (1998). Evolution of a world model for a miniature robot using genetic programming. *Robotics and Autonomous Systems*, *25*, 10–16.

Olmer, M., Nordin, P., & Banzhaf, W. (1996). Evolving real-time behavioral modules for a robot with GP. In *Proceedings of the 6th International Symposium on Robotics and Manufacturing* (pp. 67–80). New York: ASME Press.

Ostergaard, E. & Lund, H. (2003). Co-evolving complex robot behavior. In *ICES'03, The 5th International Conference on Evolvable Systems: From Biology to Hardware*. Berlin: Springer.

Parker, G. (1998). Generating arachnid gaits with cyclic genetic algorithms. In *Genetic programming 1998: Proceedings of the 3rd Annual Conference* (pp. 57–83). San Francisco: Morgan Kaufmann.

Parker, G. (2000a). Co-evolving model parameters for anytime learning in evolutionary robotics. *Robotics and Autonomous Systems*, *33*, 13–30.

Parker, G. (2000b). Evolving leg cycles to produce hexapod gaits. In *The world Automation Congress WAC 2000*, (pp. 250–255). Albuquerque: TSI Enterprises.

Parker, G. (2003a). Learning adaptive leg cycles using fitness biasing. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*. Piscataway, NJ: IEEE Press.

Parker, G. (2003b). Learning adaptive leg cycles using fitness biasing. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*. Piscataway, NJ: IEEE Press.

Parker, G. and Blumenthal, H. J. (2002). Punctuated anytime learning for evolving a team. In *World Automation Congress WAC'02*. Albuquerque: TSI Enterprises.

Parker., G. and LaRochelle, K. (2000). Punctuated anytime learning for evolutionary robotics. In *The world Automation Congress WAC'00*, (pp. 268–273).

Parker, G. and Mills, J. (1999). Adaptive hexapod gait control using anytime learning with fitness biasing. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'99* (pp. 519–524). San Francisco: Morgan Kaufmann.

Parker, G. & Rawlins, G. (1996). Cyclic genetic algorithms for the locomotion of hexapod robots. In *Proceedings of the World Automation Congress (WAC '96)*, *Robotic and Manufacturing Systems*, *3,* 617–622.

Ramsey, C. and Grefenstette, J. (1993). Case-based initialization of genetic algorithms. In *Genetic Algorithms: Proceedings of the Fifth International Conference (ICGA'93)*. San Francisco: Morgan Kaufmann.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Royal Aircraft Establishment Translation Toms, Trans 1122 B.F., Ministry of Aviation, Royal Aircraft Establishment, Franborough, Hants.

Rechenberg, I. (1973). *Evolutionstrategies: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog, Stuttgart.

Reynolds, C. (1994). Competition, co-evolution and the game of tag. In *Proceedings of the 4th Workshop on Artificial Life*. Cambridge, MA: MIT Press.

Salomon, R. (1996). Increasing adaptivity through evolution strategies. In *From Animals to Animats 4: Proceedings of the 4th International Conference on the Simulation of Adaptive Behavior (SAB'96)*, (pp. 411–420).

Salomon, R. (1997). The evolution of different neuronal control structures for autonomous agents. *Robotics and Autonomous Systems. Special issue: Robot Learning: the New Wave*, *647,* 1–15. Cambridge, MA: MIT Press.

Schultz, A. & Grefenstette, J. (1992). Using a genetic algorithm to learn behaviours for autonomous vehicles. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (pp. 739–749).

Schwefel, H.-P. (1977). *Numerische Optimierung von Computer-Modellen mittles der Evolutionsstrategie*. Basel: Birkhauser.

Sims, K. (1994). Evolving virtual creatures. In *Computer Graphics, Annual Conference Series*, (pp. 1–2). New York: ACM Press.

Spears, W., Jong, K. D., Back, T., Fogel, D., & de Garis, H. (1993). An overview of evolutionary computation. In *Proceedings of the 1993 Conference on Machine Learning*, (pp. 442–459). Berlin: Springer-Verlag.

Thompson, A. (1995). Evolving electronic controllers that exploit hardware resources. In *Advances in Artificial Life: Proceedings of the 3rd International Conference on Artificial Life*, (pp. 640–656). Berlin: Springer-Verlag.

Urzelai, J. & Floreano, D. (2000a). Evolutionary robotics: coping with environmental change. In *Proceedings of the genetic and evolutionary computation conference GECCO'00*, (pp. 941–948). San Francisco: Morgan Kaufmann.

Urzelai, J. & Floreano, D. (2000b). Evolutionary robots with fast adaptive behavior in new environments. In *Proceedings of the 3rd International Conference on Evolvable Systems: from Biology to Hardware*, (pp. 24–51). Berlin: Springer-Verlag.

Walker, J. (2003). *Experiments in evolutionary robotics: investigating the importance of training and lifelong adaptation by evolution*. PhD thesis, University of Wales.

Walker, J. & Wilson, M. (2002). How useful is lifelong evolution for robotics? In *From Animals to Animats: Proceedings of the seventh international conference on simulation of Adaptive Behavior (SAB'02)*, (pp. 347–348). Piscataway, NJ: IEEE Press.

Watson, R., Ficici, S., & Pollack, J. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *1999 Congress on Evolutionary Computation*, (pp. 335–342). Piscataway, NJ: IEEE Press.

Watson, R., Ficici, S., & Pollack, J. (2000). Embodied evolution: distributing an evolutionary algorithm in a population of robots. Technical Report CS-00-208, Department of Computer Science, Volen National Center for Complex Systems, Brandeis University, USA.

Wilson, M. (2000). Preston: A system for the evaluation of behaviour sequences. In Demmiris, J. and Birk, A., editors, *Interdisciplinary Approaches to Robot Learning*, Would Scientific *24(9)* 185–208. Singapore: World Scientific.

Wilson, M., King, C., & Hunt, J. (1997). Evolving hierarchical robot behaviours. *Robotics and Autonomous Systems. Special Issue on Robot Learning: The New Wave*, *22(3–4)* 215–230.

About the Authors

Joanne Walker has M.Sc. and Ph.D. degrees in Computer Science from the University of Wales at Aberystwyth (1999 and 2003). She is currently a research assistant in the Intelligent Robotics group at Aberystwyth, continuing her work using genetic and evolutionary algorithms for adaptive robots. Joanne is also involved in running the UK Biologically Inspired Robotics Network (biro-net), which is based at Aberystwyth.

Simon Garrett has a Ph.D. in Machine Learning and B.Sc. in Computer Science, both from the University of Wales, Aberystwyth. Dr. Garrett has worked in research associate positions at the University of Bristol and the University of Wales, Aberystwyth, and he is now a lecturer at Aberystwyth. His research interests include the application of stochastic and probabilistic methods to robotics and bioinformatics, and the use and development of biologically inspired methods, such as artificial immune systems. *Address*: Department of Computer Science, University of Wales, Aberystwyth SY23 3DB, Wales, UK. E-mail: Srng@aber.ac.uk

Myra Wilson has a BSc from Aberdeen University and a Ph.D. in Artificial Intelligence from Edinburgh. She is a lecturer at the University of Wales in Aberystwyth and head of the Intelligent Robotics Group there. Myra is also in charge of the Biologically Inspired Robotics Network (biro-net). Her interests include adaptive robotics and biologically inspired systems. *Address*: Department of Computer Science, University of Wales, Aberystwyth SY23 3DB, Wales, UK. E-mail: mxw@aber.ac.uk