Robotics and Autonomous Systems 57 (2009) 345–370

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Fitness functions in evolutionary robotics: A survey and analysis

Andrew L. Nelson [a,*], Gregory J. Barlow [b], Lefteris Doitsidis [c]

[a] Androtics, LLC, PO Box 44065, Tucson, AZ 85733-4065, USA
[b] The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
[c] Intelligent Systems & Robotics Laboratory, Department of Production Engineering & Management, Technical University of Crete, 73132, Hania, Greece

ARTICLE INFO

Article history:
Received 13 March 2007
Received in revised form
16 April 2008
Accepted 29 September 2008
Available online 1 November 2008

Keywords:
Evolutionary robotics
Fitness functions
Genetic algorithms
Autonomous learning robots
Artificial life

ABSTRACT

This paper surveys fitness functions used in the field of evolutionary robotics (ER). Evolutionary robotics is a field of research that applies artificial evolution to generate control systems for autonomous robots. During evolution, robots attempt to perform a given task in a given environment. The controllers in the better performing robots are selected, altered and propagated to perform the task again in an iterative process that mimics some aspects of natural evolution. A key component of this process – one might argue, *the* key component – is the measurement of fitness in the evolving controllers. ER is one of a host of machine learning methods that rely on interaction with, and feedback from, a complex dynamic environment to drive synthesis of controllers for autonomous agents. These methods have the potential to lead to the development of robots that can adapt to uncharacterized environments and which may be able to perform tasks that human designers do not completely understand. In order to achieve this, issues regarding fitness evaluation must be addressed. In this paper we survey current ER research and focus on work that involved real robots. The surveyed research is organized according to the degree of *a priori* knowledge used to formulate the various fitness functions employed during evolution. The underlying motivation for this is to identify methods that allow the development of the greatest degree of novel control, while requiring the minimum amount of *a priori* task knowledge from the designer.
© 2008 Elsevier B.V. All rights reserved.

1. Introduction

The primary goal of evolutionary robotics (ER) is to develop methods for automatically synthesizing intelligent autonomous robot systems. Although the greater part of current research is applied to control systems alone, ER also applies this ideal of automatic design to the creation of robot bodies (morphology) and also to the simultaneous evolution of robot control and morphology. This is often stated in terms of co-evolution of body and mind.

Automatic robot controller development methods that do not require hand coding or in-depth human knowledge are potentially of great value because it may be possible to apply them to domains in which humans have insufficient knowledge to develop adequate controllers directly. Advanced autonomous robots may someday be required to negotiate environments and situations that their designers had not anticipated. The future designers of these robots may not have adequate expertise to provide appropriate control algorithms in the case that an unforeseen situation is encountered in a remote environment in which a robot cannot be accessed. It is not always practical or even possible to define every aspect of an autonomous robot's environment, or to give a tractable dynamical systems-level description of the task the robot is to perform. The robot must have the ability to learn control without human supervision.

In contrast to intelligent autonomous mobile robots, most industrial robots perform precisely defined tasks, using methods that are well defined at a low level. For example, an industrial robot (even a very complex one) is usually described by a dynamical model, and the task it is intended to perform can be achieved by a well-defined method or procedure. Often, the task itself can also be described by a dynamical model. Arriving at a mathematical description of an optimal or near-optimal control strategy to perform the task becomes a matter of mathematical and sometimes heuristic optimization of well-defined procedures [91].

The situation is quite different for autonomous robots that must interact dynamically with complex environments. While the overall task may remain well defined at a high level, an effective solution algorithm is usually not well defined. Most non-trivial tasks for intelligent autonomous robots cannot be described adequately by tractable dynamical models. Essentially, autonomous robot control designers know what task they want a given robot to perform, but they do not know how the robot will perform the task.

* Corresponding author. Tel.: +1 520 822 6921.
E-mail addresses: alnelson@ieee.org (A.L. Nelson), gjb@cmu.edu (G.J. Barlow), ldoitsidis@dpem.tuc.gr (L. Doitsidis).

0921-8890/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2008.09.009

Control systems for autonomous robots are often programmed directly by researchers or designers. Such control programs can be very complex. Researchers must anticipate which abilities a given robot will need, and then formulate these into a control program or control hierarchy. Many researchers in the field of autonomous robot control rely on sophisticated control architectures to facilitate overall control design [88,89].

As the complexity of an environment and task for a given autonomous robot increases, the difficulty of designing an adequate control system by hand becomes a limiting factor in the degree of functional complexity that can be achieved. A potential solution to this problem is to develop methods that allow robots to learn how to perform complex tasks automatically. Developing machine learning methods for use in robotic systems has in fact become a major focus of contemporary autonomous robotics research. Some of these methods, including evolutionary robotics, focus on the ground-up learning of complete control systems. The goal of these methods is to learn the entirety of the control structure, rather than simply learning particular components, such as object classification or instances of path planning.

Learning intelligent control for autonomous agents is in some ways very different from other forms of machine learning or optimization (see [90] for an introduction to machine learning). In particular, it is often not possible to generate a succinct training data set that might be used to train controllers using batch methods or error back propagation. Defining discrete states for complex autonomous robot-environment systems is also problematic and traditional temporal difference (TD) methods such as Q-learning are not easily applied to intelligent autonomous control learning problems in dynamic continuous environments. Evolutionary robotics approaches the problem of intelligent control learning by applying population-based artificial evolution to evolve robot control systems directly. This evolutionary process represents a form of machine learning that does not necessarily require complete knowledge of environment, robot morphology, or task dynamics.

The field of evolutionary robotics is situated within a broader area of research focused on automatic methods of environment-based learning and autonomous systems development. This broader area of inquiry includes developmental robotics [108,109], artificial life, and a variety of other non-evolutionary computation-based machine learning specialties applied to fully autonomous systems. Although this survey focuses specifically on objective functions used in evolutionary robotics research, objective functions are a central component of many control learning methods applied to intelligent autonomous agents.

The field of automatic intelligent control learning for autonomous robots is in its infancy. Much of the research surveyed in this paper focused on learning how to perform relatively simple tasks. Phototaxis, for instance, is a well-studied task in ER and is representative of the complexity of tasks studied in much of the current and past research. To perform this task, a robot in an environment must identify and home in on a light source.

The current focus of ER is on developing methods for evolving controllers capable of performing more difficult and complex tasks, rather than optimizing the evolution process for tasks that have already been achieved. Hence, producing a system that could generate efficient controllers for the task of phototaxis using 10% or even 50% less computing time would not be considered a real advancement in the field. On the other hand, one particular ER effort might be considered an improvement over an earlier work if the later work required the use of much less *a priori* knowledge on the part of the researchers to evolve controllers for a similar task. In this case, the later system would have learned a greater portion of novel intelligent control, and would represent an improvement in methodology [106].

In general, the research papers reviewed in this survey report the successful evolution of controllers capable of performing the intended tasks. Moreover, most attempted research that failed to produce functional controllers will likely not have been published. Hence, the success of research is measured in the difficulty of tasks investigated, and the amount of *a priori* information needed to generate successful evolution of controllers capable of performing those tasks.

1.1. Prior work

The field of ER has been reviewed in various publications [1–8]. However, there is no current comprehensive review of the field that investigates the central issue of fitness selection methods in evolutionary robotics.

[1,3] both provide excellent reviews of the state of the field of ER in the mid-1990's. Robot controller hardware evolution is reviewed in [7] and an extensive review of the use of multi-objective optimization in evolutionary robotics is found in [8].

[5] explores issues related to training phase learning, lifetime learning and embodied learning in real robots, but that work differs considerably from our work both in focus and coverage. We focus on the issues of fitness determination and objective function formulation, and compare reported fitness functions using a common function nomenclature and classification system.

An important unanswered question within the field of ER is whether the methods used so far to obtain the moderately complex proof-of-concept results reported over the last decade and a half can be generalized to produce more sophisticated autonomous robot control systems.

1.2. Overview of robot controller evolution

In this paper, the term *controller* is used to describe the computational portion of an autonomous mobile robot system that receives information from the robot's sensors, processes this information, and produces actuator or motor commands that cause the robot to move or interact with its environment. The controller in this sense might be thought of as the brain of the robot, and some ER researchers use this terminology. In the broader field of autonomous robotics, control learning may focus on selected portions of a robot's control abilities, such as object recognition [94], path planning and localization [92,93], or error and fault accommodation. In contrast, ER research is typically directed toward learning (or evolving) the entire control system.

In ER, the process of controller evolution consists of repeating cycles of controller fitness evaluation and selection that are roughly analogous to generations in natural evolution. During each cycle, or generation, individual controllers taken from a large population of controllers attempt to perform a task or engage in some form of an evaluation period. This involves instantiating each controller into a robot (either real or simulated) and allowing the robot to interact with its environment (which may include other robots) for a period of time. In later discussions we will refer to this as an *evaluation, trial* or *test period*. Following this period, each robot controller's performance is evaluated based on a fitness function (also called an objective function). In the final step of every cycle, a genetic algorithm (GA) is applied [95]. The GA uses information generated by the fitness function to select and propagate the fittest individuals in the current population of controllers to the next generation population. During propagation, controllers are altered using stochastic genetic operators such as mutation and crossover to produce offspring that make up the next generation of controllers. Cycles are repeated for many generations to train populations of robot controllers to perform a given task,

and evolution is terminated when suitably functional controllers arise in the population.

The success of the entire process depends on how effective the fitness function is at selecting the best controllers, and it is this feature of evolutionary robotics on which we focus our attention. In this paper we survey the current ER literature with an eye towards fitness functions and the relationship between fitness evaluation methods and complexity of behavior evolved. We present a taxonomic classification of fitness functions used in evolutionary robotics research (Section 2) and use this to organize the surveyed work (Section 3).

Many variations on standard GAs are used in ER, but the majority of the research uses the traditional set of process steps consisting of test, evaluate fitness, select, mutate/recombine and propagate during each generation. Other related population-based algorithms include particle swarms, ant optimization [101] and artificial immune optimization methods [102]. Such methods incorporate dynamics observed in nature into search algorithms based on the assumption that search-algorithm-like processes observed in nature represent efficient methods honed by evolution during the course of evolution of life on Earth. These methods, as well as single agent learning methods, are also fitness function driven.

The case can be made that most forms of learning of intelligent behavior based on interaction between agent and environment share similar underlying characteristics. The main motivation for using artificial evolution and GAs in learning robots is to accommodate the computationally intractable uncharacterized high-dimension real-valued search spaces encountered in intelligent control learning problems.

1.3. The fitness function

Successful evolution of intelligent autonomous robot controllers is ultimately dependent on the formulation of suitable fitness functions that are capable of selecting for successful behaviors without specifying the low-level implementation details of those behaviors.

The fitness function is at the heart of an evolutionary computing application. It is responsible for determining which solutions (controllers in the case of ER) within a population are better at solving the particular problem at hand. In work attempting to evolve autonomous robot controllers capable of performing complex tasks, the fitness function is often the limiting factor in achievable controller quality. This limit is usually manifested by a plateau in fitness evaluation in later generations, and indicates that the fitness function is no longer able to detect fitness differences between individuals in the evolving population.

Although developing an experimental research platform capable of supporting the evolutionary training of autonomous robots remains a non-trivial task, many of the initial concerns and criticisms regarding embodiment and transference from simulated to real robots have been addressed. There are sufficient examples of evolutionary robotics research platforms that have successfully demonstrated the production of working controllers in real robots [9–12]. Also, there have been numerous examples of successful evolution of controllers in simulation with transfer to real robots [13–19]. One of the major achievements of the field of ER as a whole is that it has demonstrated that sophisticated evolvable robot control structures (such as neural networks) can be trained to produce functional behaviors in real (embodied) autonomous robots. What has not been shown is that ER methods can be extended to generate robot controllers capable of complex autonomous behaviors. In particular, no ER work has yet shown that it is possible to evolve complex controllers in the general case or for generalized tasks.

Concerns related to fitness evaluation remain largely unresolved. Much of the ER research presented in the literature employs some form of hand-formulated, task-specific fitness function that more or less defines how to achieve the intended task or behavior. The most complex evolved behaviors to date consist of three or four coordinated fundamental sub-behaviors [14,20–22]. In [14], the fitness evaluation method used was relatively selective for an *a priori*, known or predefined solution. In [20–22] the fitness functions used for selection contained relatively little *a priori* knowledge, and allowed evolution to proceed in a relatively unbiased manner. This is an interesting contrast to much of the work aimed at evolving simple homing or object avoidance behaviors, many of which use complex fitness functions that heavily bias the evolved controllers toward an *a priori* known solution.

1.4. Robots

A wide variety of robots equipped with different kinds of sensor types are used in ER. Almost all of these are mobile robots of one form or another and include wheeled mobile robots, legged robots, and flying robots.

The most typical robots used in this field are small (between 5 and 20 cm in diameter) differential drive (skid steering) robots equipped with several IR proximity sensors, photodetectors, and tactical sensors. Some of these robots also use video. For most of the work discussed in this survey, robots operate in small arenas that contain obstacles and sometimes other robots. These arenas might be small enough to be placed on a desktop, or they might be constructed on a portion of floor space in a research lab or office.

There are several robot platforms that are commercially available. The Khepera robot platform is one of the most commonly used small differential drive robot systems in ER [97]. It is of modular design and can be equipped with IR, tactile and photosensors. A CCD camera unit and gripper unit are also available. The Khepera is 5 cm in diameter, has limited computational power and is often operated via an umbilical by a remote computer. The Koala is a larger differential drive robot (30 cm in length) also manufactured by the makers of the Khepera, and has been used in a few ER experiments. Commercially available LEGO Mindstorm-based robots have also been used for several ER experiments.

Many researchers use small custom robots of their own construction for ER work. For example, the EvBot [96] is a small differential drive robot that has been used by several research groups for a variety of ER experiments.

Larger lab robots such as the RWI B21 [50] and the Nomad [28] have been used in a few ER research efforts. Unlike the smaller robots, these robots are heavy, more powerful, and capable of damaging walls and other laboratory equipment. In addition, these robots can be quite expensive and difficult to maintain.

A smaller but considerable amount of work has been done using legged robots from bipeds to octopods. The majority of these robots are custom-built by the various researchers and labs using hobby servos. In addition to these, the commercially available Sony AIBO quadruped robot has been used in a number of gait learning and locomotion learning ER experiments. This is a small 18 degree-of-freedom (DOF) robot that uses video and IR sensors.

When discussing particular research examples in the survey portion of this paper we mention briefly the type of robot used and the sensor configuration, but do not go into detail unless the robot platform is significantly different from the common differential drive systems used by the majority of the researchers.

1.5. Controller architectures

Learning control is common to all ER work. A variety of controller architectures are used in ER. These include neural networks, evolvable programs, various parameterized control structures, and evolvable hardware devices.

Neural networks are well suited for training with evolutionary computing-based methods because they can be represented by a concise set of tunable parameters. A wide variety of neural network structures have been used. The most common of these are layered feedforward or recurrent network architectures. A few of the papers cited here use Hebbian networks or other self-training networks, and these are pointed out. Neural networks are used in approximately 40% of ER work.

Evolvable programming structures are used in about 30% of the ER research. The process is referred to as genetic programming (GP). The work using evolvable hardware generally implemented some form of genetic programming or evolvable logic structure in hardware.

Much of the ER work that focused on evolution of gaits for legged robots simply evolved sets of gait control parameters. For instance, the Sony AIBO robot's gait is controlled by a set of timing and joint-position parameters, and in several of the works surveyed here, a subset of these were evolved directly. Evolving parameters of an otherwise specified gait control program differs from the majority of other ER work in that the full control system was not evolved. Most other ER work focuses on learning of monolithic control systems that act directly on sensor inputs and produce actuator commands.

1.6. Tasks and behaviors

In this subsection we will briefly discuss some of the most common tasks that robots have been evolved to perform in ER research. Some of these tasks have been studied by many different researchers over the last two decades.

Locomotion and object avoidance is one of the most frequently investigated robot tasks studied in ER. In this task robots must evolve to travel about an environment while avoiding stationary and sometimes mobile obstacles. This task might also be referred to as navigation, although technically, the term navigation generally involves traveling to and from specified locations, not just moving about without hitting anything.

Gait learning in legged robots is another commonly studied task. In the simplest form of gait evolution, functional forward locomotion is the only goal and no sensor inputs are used. Gait learning is a form of locomotion learning, but it might be considered a somewhat more difficult problem in legged robots than in wheeled robots. For locomotion to occur in wheeled robots, the wheel actuators must simply be turned on. For differential drive robots, this essentially consists of a 2-DOF system and evolving a controller to produce straight motion in an open environment would be considered trivial by modern standards. Locomotion in legged robots, on the other hand, is much less trivial. Most legged robots have between 12 and 20 DOF. Simply energizing the actuators is very unlikely to produce efficient locomotion. The leg actuators must be cycled on and off in a coordinated and controlled fashion.

Phototaxis is another frequently studied task. As mentioned earlier in the paper, robots must detect and home in on a light source in this task. Goal homing is a related task, but here, the goal location is not marked by a light source, and might not be marked at all. Environment complexity can play a significant role in the difficulty of the behavior to be learned in both goal homing and phototaxis tasks. Environments that contain objects that occlude the goal or light location from the sensors of the robots will require a more sophisticated strategy to negotiate than would be required in a simple environment.

Searching tasks are also commonly studied in ER. In searching tasks, robots travel about an environment searching for various objects. This might be considered a variation on goal homing, but the environment could contain many search objects.

Foraging is similar to searching, but the robots are also required to pick up or acquire the objects, and in some cases to then deposit the objects at a goal location. Foraging with object deposition (or object carrying) is on the complex end of the scale for tasks and behaviors studied in ER. Robots must find objects in an environment, then pick them up and carry them to another location and deposit them. These steps taken together contain an element of sequencing and cannot easily be performed by a purely reflexive system.

Predator and prey tasks involve one robot learning (or evolving) to capture another robot while the other learns to evade the first. There are several variations on this theme. Most common is a setup in which only one of the robots uses evolving controllers while the other uses a fixed hand-designed controller.

There are a few examples of other complicated tasks found in the literature. These include multiple goal homing, in which a robot must travel to two or more goal locations in a specified sequence. Another more complex task is represented by groups of robots competing against one another to find hidden objects.

1.7. Fitness landscapes

The analysis of fitness landscapes is usually considered to be an important issue in evolutionary computing applications. For a given search space, a given fitness function will define a fitness landscape or manifold. In evolutionary robotics, the search space is defined by the genome defining the controller representation (or controller and morphology representation, if body and mind are being co-evolved). Each evolvable parameter of the genome defines a dimension of the search space, and the fitness landscape is then given by the manifold defined by the fitness of each point in the search space.

In many areas of evolution computing, great effort is made to elucidate the properties of the search space and the topology of a given fitness landscape generated by application of a given fitness function. Certain more tractable fitness landscapes are amenable to specialized algorithms that may reduce computation effort, guarantee convergence or otherwise produce desirable features.

However, in ER, genome search spaces and fitness landscapes are often very difficult to characterize to the degree that significant benefit can be gained. The topologies of search spaces traversed by the evolving dynamic controller populations are generally rugged in the extreme, may have varying numbers of dimensions, and may potentially be non-static [98]. Because of these factors, search spaces and associated fitness landscapes in ER are often intractable in terms of full characterization. This state reflects the fact that the genomes are designed to be able to represent autonomous dynamic agents, at least in terms of control.

Currently, there is no adequate theory that can relate salient features of intelligent systems to representations. For example, it is difficult or impossible to distinguish between a well trained and a poorly trained neural network by any means other than direct testing. The intractable nature of fitness landscapes is one of the defining features of ER and any form of autonomous control learning based on interaction between agent and environment. Because of this underlying intractability, there is no great emphasis on fitness landscape analysis in ER. Further, and perhaps more importantly, attempts to make search spaces more tractable often impose biases into the evolving systems that reflect the designer's intuitive *a priori* knowledge of known solutions, thus reducing the system's ability to discover novel solutions.

Table 1
Fitness function classes.

Fitness function class	*A Priori* knowledge incorporated
Training data fitness functions (for use with training data sets)	Very high
Behavioral fitness functions	High
Functional incremental fitness functions	Moderate-high
Tailored fitness functions	Moderate
Environmental incremental fitness functions	Moderate
Competitive and co-competitive selection	Very low-moderate
Aggregate fitness functions	Very low

2. Classification of fitness functions in evolutionary robotics

In this section, we present a classification system for fitness functions and review current methods used for controller fitness evaluation in evolutionary robotics. The classification hierarchy is based on the degree of *a priori* knowledge that is reflected in the fitness functions used to evolve behaviors or task performance abilities. The justification for using *a priori* knowledge as a basis for classification and organization of the research is that it reflects the level of truly novel learning that has been accomplished [106]. There are of course other means by which designers introduce their own *a priori* knowledge of task solutions into the design of experimental systems intended to study evolution (or learning) in autonomous robots. These include selection of appropriate sensors and actuators, design of training environments, and choice of initial conditions. Although these other forms of introduced *a prior* knowledge are also important (and perhaps worthy of a meta-study), it is the fitness function that contains the most explicit and varied forms of task solutions knowledge. Many of the research platforms have at least qualitative commonalities of sensor capabilities and actuator arrangements. For example in more than half of the literature survived in this review, wheeled robots that employed differential drive for steering were used.

We define seven broad classes of fitness functions. These are listed in Table 1. The characteristics of each class will be discussed in this section and a full survey of ER research in terms of particular fitness functions will follow in Section 3.

2.1. Training data fitness functions

The first class of fitness functions, those used with data sets, is not exclusive to evolutionary computing methods. Training data fitness functions are used in gradient descent training methods such as error back propagation for training neural networks, and various curve-fitting and numerical methods. Here, fitness is maximized when the system in question produces a minimum output error when presented with a given set of inputs with a known set of optimal associated outputs.

For a given problem, a training data set must include sufficient examples such that the learning system can extrapolate a valid generalizable control law. Thus, at least implicitly, an ideal training data set contains knowledge of *all* salient features of the control problem in question. For controllers that are intended to perform a complex behavior or task, sufficient training data sets are usually unavailable, and the knowledge needed to create such a data set could be used to formulate a more traditional controller. The main use of training data fitness functions in autonomous control learning is in the area of mimetic learning, where a robotic system learns to mimic behavior generated by a human or other trainer. In some sense, training data fitness functions require complete *a priori* knowledge of the task to be performed, at least insofar as it is possible to generate a suitable training data set. Robots trained with such data sets learn to duplicate an *a priori* known set of system inputs and outputs. Knowledge-based training and examples of the use of training data fitness functions in ER can be found in [23–25].

2.2. Behavioral fitness functions

Behavioral fitness functions are task-specific hand-formulated functions that measure various aspects of what a robot is doing and how it is doing it. These types of functions generally include several sub-functions or terms that are combined into a weighted sum or product. These sub-functions or terms are intended to measure simple action-response behaviors, low-level sensor-actuator mappings, or other events/features local to the robot. These will be referred to as *behavioral terms*, and measure some aspect of how a robot is acting (behaving), not what it has accomplished. In contrast, *aggregate terms* measure some aspect of what the robot has accomplished, without regard to how it was accomplished.

The quality that unifies functions in the class of behavioral fitness functions is that they are made up only of terms or components that select for behavioral features of a presupposed solution to a given task. For example, if one wished to evolve robots to move about an environment and avoid obstacles, one might include a term in the fitness selection function that is maximized if a robot turns when its forward sensors are stimulated at close range. In this case the system is set up such that robots will evolve to produce a certain actuator output in response to a given sensor input. Now, selection occurs for a behavior that the designer believes will produce the effect of obstacle avoidance, but the robots are not evolving to avoid objects *per se*, they are learning to turn when their forward sensors are stimulated. This is more specific than just selecting for robots that do not collide with objects.

Some terms in a behavioral fitness function are not selective for a precise sensor-to-actuator mapping, but rather for a desired control feature. For example, if one wished to evolve a robot controller that spent most of its time moving, one might include a term in the fitness function that is maximized when forward motion commands result in continued forward motion of the robot over time (if the front of a robot were in contact with an immobile object, it would not move forward regardless of its current actuator commands). This example term is not selective for an exact sensor-to-actuator mapping. There are other possible formulations that could also produce the desired control feature, such as a term that maximized the ratio of forward motion to forward sensor activity. Hence, this type of term does not require quite as much *a priori* knowledge of the exact details of the control law to be learned. Examples of the use of behavioral fitness functions can be found in [13,26,27].

2.3. Functional incremental fitness functions

Functional incremental fitness functions begin the evolutionary process by selecting for a simple ability upon which a more complex overall behavior can be built. Once the simple ability is evolved, the fitness function is altered or augmented to select for a more complex behavior. This sequence of evolution followed by fitness function augmentation continues until eventually the desired final behavior is achieved. The overall process can be considered one of explicit training for simple sub-behaviors followed by training for successively more complex behaviors. Often, an artificial evolution process that makes use of an incremental fitness function is referred to as *incremental evolution*.

Functional incremental fitness functions address a major difficulty in evolutionary robotics. For difficult tasks, it is possible

that some or all of the controllers in a newly initialized population will possess no detectable level of ability to complete the task. Such a controller is referred to as being sub-minimally competent. If all the controllers in an initial population of controllers are sub-minimally competent for a particular fitness function, then the fitness function can generate no selective pressure and the population will fail to evolve. Functional incremental fitness functions overcome the problem of sub-minimally competent controller populations by augmenting the difficulty of the task for which the controllers are being evolved during the course of evolution.

One main criticism of using functional incremental fitness functions is that they may restrict the course of evolution to the degree that resulting controllers cannot be considered to have evolved truly novel behaviors. The designer is not only responsible for including features of a desired solution (as is the case for tailored fitness functions, discussed in the next subsection), but must also structure the search path through the controller's configuration space (search space). For non-trivial robot control tasks, there is no guarantee that this design problem is tractable. Examples of ER research making use of functional incremental fitness functions are found in [9,33,34].

2.4. Tailored fitness functions

In addition to behavior-measuring terms, tailored fitness functions contain aggregate terms that measure some degree or aspect of task completion that is divorced from any particular behavior or method. Hence, tailored fitness functions combine elements from behavioral fitness functions and aggregate fitness functions (discussed in Section 2.7 of this section). As an example, suppose a phototaxis behavior is to be evolved. A possible fitness function might contain a term that rewards a controller that arrives at the light source by any means, regardless of the specific sensor-actuator behaviors used to perform the task. This term would be considered an aggregate term. If it were the only term in the fitness function, then the whole function would be considered aggregate. If the function also contained a second behavioral term, for example, one that maximized the amount of time the robot spent pointing toward the light source, then the two terms together would constitute an example of a tailored fitness function. Note that this second term, selecting for pointing toward the light source, does represent implicit assumptions about the structure of the environment and may not be the best way to approach the light source in some complex environments.

Unlike true aggregate fitness functions, aggregate terms in tailored fitness functions may measure a degree of partial task completion in a way that injects some level of *a priori* information into the evolving controller. For example, in the phototaxis task, a tailored fitness function might contain a term that provides a scaled value depending on how close the robot came to the light source. This may seem at first glance to be free of *a priori* task solution knowledge, but it contains the information that being closer to the goal is inherently better. In an environment composed of many walls and corridors, linear distance might not be a good measure of fitness of a given robot controller. We use the term "tailored" to emphasize that these types of fitness functions are task-specific hand-formulated functions that contain various types of selection metrics, fitted or *tailored* by the designer to accommodate the given problem, and often contain solution information implicitly or explicitly. Examples of work using tailored fitness functions can be found in [28–30].

Together, tailored fitness functions and behavioral fitness functions make up by far the largest group of fitness functions used in current and past evolutionary robotics research. These types of fitness functions are formulated by trial and error based on the human designer's expertise.

2.5. Environmental incremental fitness functions

Rather than simply increasing the complexity of the fitness selection function, one form of incremental evolution involves augmenting the difficulty of the environment in which the robots must operate. This is referred to as *Environmental incremental* evolution. Environmental incremental evolution may not constrain the controller's search space to the degree that evolution must converge on a particular predefined solution. Relatively little work has been done using environmental incremental evolution. In [35] the authors used Environmental incremental selection to evolve controllers for a fairly complex peg collection task. That research showed that Environmental incremental evolution can produce robot controllers capable of expressing complex behaviors. However, it is not clear to what degree the selection and augmentation of training environments shaped the final evolved controllers. Other examples include [36,37,83].

2.6. Competitive and co-competitive fitness selection

Competitive fitness selection utilizes direct competition between members of an evolving population. Controllers in almost all ER research compete in the sense that their calculated fitness levels are compared during selection and propagation. However, in competitive evolution robot controllers compete against one another within the same environment so that the behavior of one robot directly influences the behavior, and therefore fitness evaluation, of another. For example, in a competitive goal-seeking task, one robot might keep another from performing its task by pushing it away from the goal. Here, the second robot might have received a higher fitness rating if it had not been obstructed by the first robot. Examples of the use of intra-population competition, in which the fitness of individual robots were directly affected by interaction with other robots using controllers from the same population, has been investigated in [21].

In co-competitive evolution two separate populations (performing distinct tasks) compete against each other within the same environment. Examples of co-competitive evolution involving populations of predator and prey robots exist in the literature [10,38,39,84]. Two co-evolving populations, if initialized simultaneously, stand a good chance of promoting the evolution of more complex behaviors in one another. As one population evolves greater skills, the other responds by evolving reciprocally more competent behaviors. [84] discusses this putative explanation for the selective power of competitive selection, termed the Red Queen Effect. The changing behavior of the evolving competing agents alters the fitness landscape, essentially generating a more and more arduous selection criterion but without changing the fitness function explicitly. The research presented in [10,38,39] shows this effect in evolving robot controller populations to a degree, but results from other areas of evolutionary computing suggest that given the correct evolutionary conditions, aggregate selection combined with intra-population competition within a single population performing a directly competitive task can produce very competent systems [40,41].

2.7. Aggregate fitness functions

Aggregate fitness functions select only for high-level success or failure to complete a task without regard to how the task was completed. This type of selection reduces injection of human bias into the evolving system by aggregating the evaluation of benefit (or deficit) of all of the robot's behaviors into a single success/failure term. This is sometimes called *all-in-one* evaluation. Examples of aggregate fitness selection are found in [17,16,32].

Consider the following foraging task: a robot is to locate and collect objects and then deposit them at a particular location (or a "nest"). An aggregate fitness function would contain information only related to task completion. Suppose the task is considered to be complete when an object is deposited at the nest. An example of an aggregate fitness function for this task would be one that counted the number of objects at the nest after the end of a trial period.

Until recently, aggregate fitness selection was largely dismissed by the ER community. This is because initial populations of controllers generally have no detectable level of overall competence to perform non-trivial tasks (i.e. they are *sub-minimally competent*). In the example above, if the objects were sparsely distributed in a complex environment, and the controllers in the initial un-evolved population were truly randomly configured without any navigation, object recognition or homing abilities, the chances of one of them completing the task by chance are diminishingly small. This situation is often referred to as the *bootstrap problem* [31]. Completely aggregate selection produces no selective pressure in sub-minimally competent populations at the beginning of evolution and hence the process cannot get started.

Even so, aggregate fitness selection in one form or another appears to be necessary in order to generate complex controllers in the general case if one is to avoid injecting restrictive levels of human or designer bias into the resulting evolved controllers. For the evolution of truly complex behaviors, selection using behavioral fitness functions and incremental fitness functions results mainly in the optimization of human-designed controller strategies, as opposed to the evolution or learning of novel intelligent behavior.

It is possible to overcome some of the problems associated with aggregate selection. One approach is to use a tailored fitness function to train robots to the point at which they have at least the possibility of achieving a given complex task at some poor but detectable level, and then to apply aggregate success/failure selection in conjunction with intra-population competition to drive the evolutionary process to develop competent controllers. Intra-population competition presents a continually increasing task difficulty to an evolving population of controllers and may be able to generate controllers that have not been envisioned by human designers.

The chart in Fig. 1 relates the classes of fitness functions to degrees of *a priori* knowledge incorporated into evolving populations of robot controllers. The chart is qualitative and reflects general associations. Some of the fitness function classes discussed previously can be formulated to incorporate varying degrees of *a priori* knowledge into evolving populations and are depicted spanning several levels on the horizontal axis.

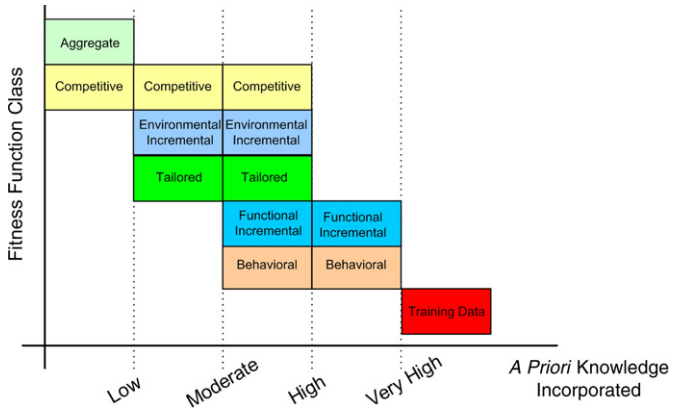

Fig. 1. Chart relating classes of fitness functions to levels of incorporated *a priori* knowledge.

3. A survey of fitness evaluation functions in ER

In this section we survey current and past ER research and organize the work using the classification system presented in Section 2. The surveyed research is listed by fitness function class and by date in Tables 3–8. A distinction is made between work that involved real robots and that in which only simulated robots were used. We have endeavored to reference most of the major research efforts that involved real robots at some level or another. Some work that was conducted only in simulation but never tested on real robots is also discussed at the end of this section. Here, though, we do not attempt a comprehensive summary of the purely simulated work.

Before we continue into our main survey and discussion of fitness functions used in ER, we will lay out some general bounds, define conventions used for symbolic representation of fitness functions, and define features or elements that are common to most of the reviewed research.

Almost all of the work considered in this survey employed some form of population-based artificial evolution in which the candidate solutions being evolved are autonomous robot controllers. Although the evolutionary algorithms vary to a degree from work to work, most of them fall within a general class of stochastic hill-climbing learning algorithms. Unless otherwise stated, the research papers reviewed here may be assumed to use an evolutionary method roughly equivalent to that which was outlined in the introduction to this paper. Population sizes vary widely. Much of the research used population sizes in the range of 20–100. Twenty to 300 generations are generally reported to be required to evolve suitable controllers for the majority of tasks investigated, but generations might range into the thousands in some cases. In a few cases the evolutionary algorithms differ significantly from standard forms and in these cases a short description of the artificial evolution methods used will be included.

As noted in the introduction, efficacy of methods beyond that of obtaining reasonably functional controllers is not a primary focus of evolutionary robotics in its current state of development. The focus, rather, is upon designing evolutionary systems able to evolve controllers capable of performing new tasks of greater complexity. It is true that some methods or robot platforms may show a 2-fold (or even 10-fold) increase in training efficiency over others, but it is the fitness function that finally determines the achievable performance.

We have endeavored to translate the diverse formulations of the fitness functions into a consistent summary representation. It is important to note that in some cases details of the original fitness functions are abstracted so that the underlying forms can be presented and compared in a standardized way. In some cases, fitness functions have been generated from a text description. This is done so that the underlying natures of functions can be compared more directly.

Where possible, f will be used to indicate instantaneous fitness. In general, fitness is maximized during an artificial evolution process, and unless otherwise stated, it will be assumed that a given fitness function is optimized when it is maximized. Those functions that are minimized are denoted with a minus sign subscript ($f_{(-)}$).

In the process of evaluating the fitness of a given robot controller, fitness functions are commonly integrated or averaged with respect to sensory-motor cycles or time steps over the course of a fitness evaluation trial period. In many cases, researchers report fitness functions that explicitly include this integration process. To facilitate comparison, and to define a simple unified

Table 2
List of common symbols used in fitness function representation.

Symbol	Meaning
F	Explicit fitness function
f	Fitness function integrand or summand
φ	Non-standard integrand or summand
f_1, f_2, f_3	Incremental fitness function integrand
d	Distance traveled
v	Velocity (drive motor)
s	Sensor activation level
B	Boolean sub-function
c	Constant coefficient

format as much as possible, the standardized representation used in this survey presents fitness functions in the form of an integrand only. Integrals or averaging summations are not explicitly symbolized in the standardized representation of these fitness functions. Integration or summing is assumed to be part of the evaluation process and is common to almost all the work surveyed. This means that for a particular fitness function integrand reported as $f(\cdot)$ in this survey, the actual fitness calculation for an evaluation period of time length N (or of N time steps) would be calculated by

$$F(t) = \frac{1}{N} \int_{t_0}^{t_N} f(t, \mathbf{q}(t)) \mathrm{d}t \tag{1}$$

or

$$F(t) = \frac{1}{N} \sum_{t_0}^{t_N} f(t, \mathbf{q}(t)) \tag{2}$$

where t, t_0, t_N, and $\mathbf{q}$ represent time, initial time, final time and a vector of other (possibly implicit) functions of time respectively. Note that in many cases t does not relate directly to physical time, but rather measures time steps in a simulation environment or measures a quantized form of time. Among others, the symbol k is used in some of the referenced works to indicate discrete time, but we use the symbol t in all formulas to facilitate comparison of general forms.

Fitness functions that cannot be reduced to an average, sum or integral are stated explicitly. In these cases an uppercase F is used and represents the entire cumulative fitness of a given individual measured over a given evaluation period. Aggregate fitness functions, for example, usually only report success or failure after a given trial evaluation period and do not represent a continuous integration or summing process.

Fitness functions whose values depend on specific events, or that use secondary summations, integrals, or other terms that are not integrated with respect to time are also reported in full form. Occasionally researchers employ fitness metrics that update fitness at specific trigger points during an evaluation period rather than at each time point. These functions will be given using a lower-case phi (φ).

Common terms and factors appear in many fitness functions and where possible we will use consistent notational conventions. These include distance traveled, speed, and sensor activation levels, and these will be represented by d, v, and s respectively. Boolean functions will be represented with an uppercase B. Constant coefficients will be represented by c. In the case of incremental fitness functions, f_1, f_2, f_3 and so on will be used to indicate the functions and their order of application. Table 2 provides a list of common symbols used in this paper in the representation of fitness functions.

Tables 3–8 list the main body of work cited in this survey, and include the fitness function class used, the author(s) and year of publication of the citation, the task or behavior investigated, the environment in which the evolution was performed, the type of robot used, and the controller architecture or learning representation used.

It is our view that evolutionary robotics work should be verified in real robots. Physical verification in real robots forces researchers to use simulations that are analogous to the physical world. Some subtleties of control are contained within the robot–world interface and are easily overlooked. In particular, adequate simulators must maintain a suitable representation of the sensory-motor-world feedback loop in which robots alter their relationship to the world by moving, and thus alter their own sensory view of the world. Robotics work involving only simulation, without physical verification, should not be considered fully validated. Much of the pure-simulation work falls into the category of artificial life (AL), and many of these simulation environments include unrealistic representation or rely on sensors that report unobtainable data or that report conceptual data. That said, learning in simulation with transfer to real robots has been repeatedly demonstrated to be viable over the last decade. Much of this work has involved new physics- and sensor-based simulators. Introducing noise into the simulation environment has been shown to aid in transference of evolved controllers from simulated to real robots [27] and continues to be studied [105]. The verification of evolved controllers in real robots allows a clear distinction to be made between the large amount of work done in the field of artificial life, and the similar, but physically grounded work pursued in evolutionary robotics.

We focus on providing a comprehensive survey of evolutionary robotics in terms of fitness functions used during training of controllers; later in this section we list these fitness functions explicitly by class. In order to give some context, we also include some details of the tasks learned by the robots, controller representations and other experimental details, but the review is not meant to comprehensively describe all aspects of the experimental procedures used by the researchers.

If included in the individual papers surveyed, we also report the population size and number of generations required for successful evolution of competent controllers. The reader may note that there is considerable variation in the number of generations required to evolve controllers for a given task. These differences reflect various aspects of the individual algorithms used, the evolvable controller structures used, and the physical robots involved. Beyond the fitness function we do not attempt to delve into the specifics of the genetic algorithms used unless the details are salient or differ significantly from the norm.

The relative efficiency of controllers evolved in different research efforts is not the central focus of comparison in this survey, or in the field of evolutionary robotics as a whole. Most of the cited research efforts did produce functional controllers. There are however a few works that did not produce robot controllers capable of achieving the particular tasks intended. These are pointed out when discussed.

The works surveyed are in some ways disparate, and an absolute direct quantitative comparison of the various results is not suggested. Complexity of tasks learned, as well as the amount of *a priori* information needed to select for a given task using a given robot system are central issues in ER. These two factors, task complexity and *a priori* information used during evolution, can form the basis of a qualitative comparison of various methods use to evolve controllers. The most advanced work is that which evolves controllers for the most complex tasks while minimizing the amount of *a priori* information contained in the fitness functions used to drive the evolutionary process. For the complexity of the simple tasks studied in current research, and common fitness function formulations, this is a viable general approach. However, as the state of the art of ER becomes more advanced, even qualitative comparison of different ER research efforts will require a greater emphasis on formal comparison methods based on task difficulty metrics and machine intelligence quotient definitions.

Table 3
Summary of ER research using behavioral fitness functions.

Citation	Author(s), Year of publication	Task evolved/Learned	Embodied/Real/Simulated	Robot platform	Evolved controller type/Algorithm
[11]	Floreano and Mondada, 1996	(1) Locomotion with object avoidance (2) Locomotion with periodic goal homing	Embodied	Khepera	Neural network
[13]	Lund and Miglino, 1996	Locomotion with object avoidance	Simulated, transferred to real	Khepera	Neural network
[26]	Banzhaf et al., 1997	(1) Locomotion with object avoidance (2) object following (3) wall following (4) light avoidance	Embodied	Khepera with IR sensors	Evolvable program (GP)
[27]	Jakobi, 1998	Locomotion with object avoidance	Simulated, transferred to real	Octopod robot	Neural network
[42]	Gomi and Ide, 1998	Gait evolution	Embodied	OCT-lb, Octopod robot	Set of gait control parameters
[43]	Matellán et al., 1998	Locomotion with object avoidance	Embodied	Khepera	Fuzzy logic controller
[15]	Nordin et al., 1998	Locomotion with object avoidance	Simulated, transferred to real	Khepera	Evolvable program (GP)
[44]	Liu et al., 1999	Object pushing	Embodied	Custom built robot (JUNIOR)	Evolvable program (GP)
[45]	Seok et al., 2000	Phototaxis with obstacle avoidance	Embodied	Custom built robot	Evolvable hardware (FPGA)
[46]	Ziegler and Banzhaf, 2001	Locomotion with object avoidance	Simulated, transferred to real	Khepera	Directed graph

Table 4
Summary of ER research using functional incremental fitness functions.

Citation	Author(s)/Year of publication	Task evolved/Learned	Embodied/Real/ Simulated	Robot platform	Evolved controller type/Algorithm
[9]	Harvey et al., 1994	Differential goal homing	Embodied	Gantry robot	Neural network
[33]	Lee et al., 1997	Object pushing with goal homing	Simulated, partial transfer to real	Khepera	Evolvable program (GP)
[71]	Filliat et al., 1999	Locomotion with object avoidance	Simulated, transferred to real	SECT Hexapod robot	Neural network
[36]	Pasemann et al., 2001	Goal homing with object avoidance	Simulated, transferred to real	Khepera	Neural network
[34]	Barlow et al., 2005	Goal homing and circling	Simulated, transferred to real	EvBot	Evolvable program (GP)

3.1. Training data fitness functions

Training data fitness functions such as those used in back propagation training of neural networks require full knowledge of the solution sought in the form of a training data set. As such, these functions represent a form of solution optimization and/or n-dimensional surface fitting. We mention these here for completeness and context, but these methods fall outside the focus of this review and of ER, and they cannot be used to discover intelligent control solutions whose features are not captured in a training data set (and therefore known *a priori* at some level).

In mimetic methods, a training data set is generated by recording the sensor inputs and motor outputs of a system while it is performing a particular task. Such data sets are often derived from a teleoperated system controlled by a human, and the resulting trained systems in effect learn to mimic a particular example of a human performing the task.

Also along these lines, *breeder* or *clicker* training does not use a specific training data set, but requires a human trainer to provide fitness feedback during training [99,100]. In essence, a new set of training examples is created and coded (evaluated as positive or negative) during each training session. In breeder training, the trainer need not be able to define an explicit fitness function, but he or she must still rely on his or her own *a priori* knowledge of how to perform the task which the agent is being trained to perform.

3.2. Behavioral fitness functions

Behavioral fitness functions measure fitness by measuring qualities or features of how a robot behaves while that robot is attempting to perform a task. Behavioral fitness functions do not directly measure how well the robot has accomplished its overall task *per se*. Task completion is measured implicitly by the terms that measure various aspects of the robot's behavior (see Section 2 for an illustrative example). Research that employed behavioral fitness functions is summarized in Table 3.

In [11], an experiment is discussed in which neural network-based controllers for a Khepera robot were evolved to perform a navigation and obstacle avoidance task. During the experiment, the robot (or more precisely, a population of neural networks) learned to navigate around a maze-like environment with a single closed loop, and to avoid bumping into walls while doing so. The robot was equipped with IR sensors for detection of its environment. The fitness function integrand used to select the fittest controllers during evolution was

$$f = \text{mean}(v_l, v_r)(1 - \sqrt{|v_l - v_r|})(1 - s_{ir}) \quad (3)$$

where v_l and v_r are left and right drive motor speeds, and s_{ir} is the greatest current activation level of the IR sensors. This is considered a behavioral fitness function because it bases fitness on local motor behaviors and sensor responses and does not directly

Table 5
Summary of ER research using tailored fitness functions.

Citation	Author(s), Year of publication	Task evolved/Learned	Embodied/Real/ Simulated	Robot platform	Evolved controller type/Algorithm
[47]	Hoffmann and Pfister, 1996	Goal homing with object avoidance	Simulated, transferred to real	Custom lab robot	Fuzzy logic controller
[48]	Thompson, 1996	Locomotion with object avoidance	Simulated, transferred to real	Sussex Mr. Chips robot	Evolvable hardware (FPGA)
[28]	Schultz et al., 1996	Agent herding	Simulated, transferred to real	Nomad 200	Evolvable rule set
[14]	Nolfi, 1997	Foraging with object deposition	Simulated, transferred to real	Khepera with gripper	Neural network
[29]	Keymeulen et al., 1998	Target homing with obstacle avoidance	Simulated, transferred to real	Custom lab robot	Evolvable hardware (FPGA)
[49]	Ishiguro et al., 1999	Object pushing with goal homing	Simulated, transferred to real	Khepera	Neural network
[50]	Ebner and Zell, 1999	Locomotion with object avoidance	Simulated, transferred to real	RWI B21	Evolvable program (GP)
[20]	Floreano and Urzelai, 2000	Sequential goal homing	Embodied	Khepera, Koala	Neural network
[52]	Sprinkhuizen-Kuyper et al., 2000	Object pushing	Simulated, transferred to real	Khepera	Neural network
[53]	Wolff and Nordin, 2001	Gait optimization	Embodied	ElVINA (biped)	Gait parameter set
[54]	Nehmzow, 2002	(1) photo-orientation (2) object avoidance (3) robot seeking	Embodied	Custom LEGO robots	Evolvable program (GP)
[12]	Watson et al., 2002	Phototaxis	Embodied	Custom robot	Neural network
[56]	Marocco and Floreano, 2002	Locomotion with wall avoidance	Embodied	Koala	Neural network
[57]	Okura et al., 2003	Locomotion with object avoidance	Embodied	Khepera	Evolvable hardware (FPGA)
[30]	Quinn et al., 2002	Coordinated movement	Simulated, transferred to real	Custom robots	Neural network
[58]	Gu et al., 2003	Object (ball) homing	Embodied	Sony AIBO	Evolvable fuzzy logic controller
[55]	Simões and Barone, 2004	Locomotion with object avoidance	Embodied	Custom robots	Neural network
[59]	Nelson et al., 2004	Locomotion with object avoidance	Simulated, transferred to real	EvBot	Neural network
[60]	Boeing et al., 2004	Gait evolution	Simulated, transferred to real	Andy Droid robot	Spline controller
[61]	Hornby et al., 2005	Gait evolution	Embodied	Sony AIBO	Gait parameter set
[62]	Kamio and Iba, 2005	Object pushing with goal homing	Simulated, transferred to real	Sony AIBO, HOAP-1	Evolvable program (GP)
[22]	Capi and Doya, 2005	Triple sequential goal homing	Simulated, transferred to real	Cyber Rodent	Neural network
[63]	Parker and Georgescu, 2005	Phototaxis with obstacle avoidance	Simulated, transferred to real	LEGO Mindstorm	Evolvable program (GP)
[110]	Trianni and Dorigo, 2006	Coordinated locomotion with hole avoidance	Simulated, transferred to real	Swarm-bot	Neural network

Table 6
Summary of ER research using environmental incremental fitness functions.

Citation	Author(s)/Year of publication	Task evolved/Learned	Embodied/Real/Simulated	Robot platform	Evolved controller type/Algorithm
[37]	Miglino et al., 1998	Goal homing with object avoidance	Simulated, transferred to real	Khepera	Neural network
[35]	Nakamura, 2000	Foraging with object carrying	Simulation only	Simulated Khepera	Neural network

Table 7
Summary of ER research using competitive fitness functions.

Citation	Author(s)/Year of publication	Task evolved/Learned	Embodied/Real/ Simulated	Robot platform	Evolved controller type/Algorithm
[10]	Nolfi and Floreano, 1998	Pursuit and evasion	Embodied	Khepera	Neural network
[21]	Nelson and Grant, 2006	Competitive goal homing with object avoidance	Simulated, transferred to real	EvBot	Neural network

Table 8
Summary of ER research using aggregate fitness functions.

Citation	Author(s), Year of publication	Task evolved/Learned	Embodied/Real/ Simulated	Robot platform	Evolved controller type/Algorithm
[17]	Hornby et al., 2000	Object pushing	Simulated, transferred to real	Sony AIBO	Neural network
[64]	Earon et al., 2000	Gait evolution	Embodied	Hexapod robot Kafka	Evolvable state lookup tables
[16]	Lipson and Pollack, 2000	Locomotion (co-evolution of body)	Simulated, transferred to real	Auto-fabricated modular robots	Neural network
[65]	Hornby et al., 2001	Locomotion (co-evolution of body)	Simulated, transferred to real	TinkerBot modular robots	Actuator control parameter set
[32]	Hoffmann and Montealegre, 2001	Locomotion with object avoidance	Embodied	LEGO Mindstorm	Evolvable sensor-to-motor excitation mapping
[66]	Augustsson et al., 2002	Flying lift generation	Embodied	Winged robot	Genetic programming
[67]	Zufferey et al., 2002	Locomotion with wall avoidance	Embodied	Robotic blimp	Neural network
[68]	Macinnes and Paolo, 2004	Locomotion (co-evolution of body)	Simulated, transferred to real	LEGO-servo modular robots	Neural network
[69]	Zykov et al., 2004	Gait evolution	Embodied	Pneumatic Hexapod robot	Gait parameter set
[70]	Chernova and Veloso, 2004	Gait evolution	Embodied	Sony AIBO	Gait parameter set

measure partial or overall task completion. At each generation during the evolutionary process every network in the controller population was tested on a real robot in a real environment. Evolution performed without the use of a simulator, as in this case, is referred to as embodied evolution. The researchers reported that after the 50th generation, the fittest evolved neural network-based controller performed the task at near optimum levels and was able to travel around its environment indefinitely without colliding with walls or getting stuck in corners.

A further experiment using the same platform was also discussed in [11]. Neural networks were again evolved to control a Khepera robot. The task for the robot was a periodic goal homing behavior in which the robot was to travel about an arena for a period of time and then move to a goal location and remain there for a short time. The goal location was marked by a light source and the robot was equipped with photosensors in addition to its IR sensors. The motivation for the experiment was to evolve a behavior that could allow a robot to return to a battery recharging station, hence the robot was given a simulated energy level that would fall to zero after a period of time. The fitness function integrand used was

$$f = \text{mean}(v_1, v_2)(1 - s_{ir}). \tag{4}$$

Note that a robot that recharges its virtual energy level will achieve a greater mean velocity over a long evaluation period than one that runs out of energy too far away from the recharging station. In addition, the recharge station was placed next to a wall so that robots must spend time away from it to maximize the $(1 - s_{ir})$ factor of f. As in the first experiment in [11], embodied evolution was employed. Evolution of successful controllers took 10 h of testing time with the real robot and represented 240 generations with a population of 100 controllers.

In [13], experiments on a locomotion and object avoidance task similar to the work presented in [11] were reported. Simple neural networks with no hidden layers were evolved to perform the task. The work also used the Khepera robot platform and evolution was conducted using a behavioral fitness function integrand similar to that used in [11]:

$$f = \text{mean}(v_l, v_r)(1 - (v_l - v_r)^2)(1 - s_{ir}) \tag{5}$$

where v_l and v_r are left and right drive motor speeds, and s_{ir} is the greatest current activation level of the IR sensors. Using a population of 100 neural network controllers, evolution was initially performed in a simulation environment for 200 generations, and then optimized in a real robot for an additional 20 generations. Robots with the fittest evolved neural controllers were reported to be able to perform their intended task reliably in a real environment using the real robot.

[26] evolved four separate behaviors using embodied evolution and genetic programming (GP). A Khepera equipped with 8 IR proximity sensors was used. All of the fitness functions used were behavioral and couched in terms of function regression mapping sensor inputs to actuator outputs. The fitness functions used did not use measurements of direct task completion for fitness evaluation, but rather they selected for sensory-motor behaviors that the researchers deemed would produce the ability to perform the tasks. The behaviors evolved were forward motion with object avoidance, object homing/following, wall following, and hiding in the dark (light avoiding). The four fitness function integrands used for the four behaviors follow respectively:

$$f_{(-)} = s_{ir} - (v_l + v_r - |v_l - v_r|) \tag{6}$$

where s_{ir} is the sum of the activations of all of the IR sensors and v_l, v_r are the left- and right-hand motor velocities;

$$f_{(-)} = (s_{ir1} + s_{ir2} + s_{ir3} + s_{ir4} - c)^2 \tag{7}$$

where c is a constant picked so that the robot will learn to follow a distance behind the object such that the sum of the four forward facing sensors is near c;

$$f_{(-)} = (s_{ir1} - c_1)^2 + (s_{ir2} - c_2)^2 + {s_{ir3}}^2 - (v_l + v_r)^2 \tag{8}$$

where s_{ir1} and s_{ir2} are sensor activations on the wall-side of the robot, s_{ir3} is a sensor on the outward facing side, and c_1 and c_2 are constants; and

$$f_{(-)} = s_{\text{photo}} - (v_l + v_r - |v_l - v_r|) \tag{9}$$

where s_{photo} is the activation of a photosensor. The authors report successful evolution of these four behaviors, using purely reactive and memory-based machine language GP formulations, but no specific training data were presented.

For very simple tasks, one might define the task to be exactly that which is accomplished by producing a particular sensory-motor behavior. In cases where there is no distinction between an overall task description and the low-level sensory-motor behavior, the task will be classified as behavioral.

In [27], locomotion with obstacle avoidance in legged robots was evolved. Robot controllers were evolved in a minimal simulation environment and then transferred to a real robot for

verification. The robot had IR sensors on the right- and left-hand sides and a tactile bumper on the front. A behavioral fitness function is used in this work and has four cases, each designed to calculate fitness to perform a desired aspect of the task. The author defines this in terms of an extended case statement. This can be represented as an integrand of four terms with mutually exclusive Boolean coefficients as follows:

$$f = B_1(v_l + v_r) + B_2(v_l - v_r) + B_3(-v_l + v_r) + B_4(-v_l - v_r) \tag{10}$$

where v_l and v_r are the left- and right-hand-side velocities of the robot, and B_1–B_4 are mutually exclusive Boolean coefficients that are non-zero under the following conditions: B_1 is non-zero when no obstacles are in range of the IR sensors and the bump sensor is not engaged, B_2 is non-zero when there is an object in range of the right-hand IR sensor, B_3 is non-zero when there is an object in range of the left-hand IR sensor, and B_4 is non-zero when the bump sensor is engaged. The target behavior is defined at the level of sensor readings and robot body responses in each of the cases and the fitness is formulated to select for these only. Further, evolution took place in a carefully structured minimal simulation environment in which only dynamics that the designers believed would be relevant to an optimal solution were reproduced. All other dynamics were simply structured to produce a very low fitness in the simulated robot. As with a few of the other evolutionary robotics experiments that used very solution-specific fitness functions, in this work, a novel solution cannot be considered to have been truly learned by the system. Rather, the system has been programmed in a roundabout way to reproduce a particular *a priori* known solution. Another unusual feature of this work is that 3500 generations were used to develop controllers that produced effective locomotion in the real robot. This is between 10 and 100 times the number required for most other similar reported experiments. However, it should be noted that this is one of only a handful of research efforts to investigate intelligent control learning in an octopod.

In [42], hexapod gaits were evolved using a real hexapod robot. The behavioral fitness function used was one of the most complicated found in the literature for the task of legged-robot locomotion. We only summarize it here:

$$F = (strides)(1 - overcurrent)(balanced)(1 - bellytouch) \tag{11}$$

where each of the terms is a function based on a combination of the robot's behavior and sensor inputs. The function *strides* counts leg cycle movements, *overcurrent* measures actuator commands that exceed the current capacity of the leg motors, *balanced* measures the degree of tilt of the robot body, and *bellytouch* counts the number of times the robot's body falls low enough to scrape on the floor. The hexapod was able to evolve efficient gaits within fifty generations. As is the case with several of the other gait-learning research examples, this robot had no sensors and thus did not learn to react intelligently or dynamically to the environment *per se*. In contrast to this very complex fitness function, similar examples of gait learning have been achieved using aggregate fitness functions (see Section 3.8).

A population of fuzzy logic rule-based controllers was evolved in [43]. The robot task was locomotion and object avoidance. The fitness function integrand includes a parsimony term to reduce the number of rules in the controller fuzzy rule set:

$$f = \frac{\text{mean}(v_l, v_r)(1 - |v_l - v_r|)(1 - s_{ir})}{|rules|} \tag{12}$$

where v_l and v_r are left and right drive motor speeds, s_{ir} is the greatest current activation level of the IR sensors, and |rules| counts the number of rules in the controller's fuzzy logic rule set. A population of 100 individuals was evolved for 100 generations in a real Khepera robot, and the authors report that successful controllers able to travel around their environment without colliding with obstacles are developed by the 60th generation.

In [15] genetic programming was used to evolve locomotion and object avoidance in Khepera robots. A behavioral fitness function was used and is given by

$$f = c(|v_l - v_r| + |v_r| + |v_l| - (v_l - v_r)) + \sum s_{ir} \tag{13}$$

where v_l and v_r are the left and right wheel motor speeds, and $\sum s_{ir}$ represents the sum of the activations of the proximity sensors. The authors used an extremely large population size of 10 000 and ran evolutions for 250 generations. They repeated evolution runs 100 times in simulation starting with different seed populations and reported that 82 out of the 100 runs produced useful controllers able to perform the task of obstacle avoidance while traveling about a small environment with a single circular path.

Controllers for a wall-following behavior were also evolved in [15]. A very complex conditional fitness selection method that specifies desired responses to possible sensor activation patterns was used. This fitness selection method essentially specified the solution to be evolved and injected a very high level of *a priori* knowledge into the evolved controllers.

The authors of [44] describe an object-pushing task in terms of a sumo-robot behavior. Controllers were evolved to push objects out of an arena. GP was used to evolve controllers composed of behavioral primitives such as "more forward" and "left-hand turn". The fitness function integrand used here counts the number of active sensors, the number of arms in contact with the object, and the number of arms holding the object:

$$f = \sum s_{\text{active}} + \sum arms_{\text{holding}} + \sum arms_{\text{touching}} \tag{14}$$

where $\sum s_{\text{active}}$ is the number of active proximity sensors, $\sum arms_{\text{holding}}$ is the number of arms applying side pressure to the object, and $\sum arms_{\text{touching}}$ is the number of arms in contact with the object. The researchers reported that the robot was able to learn to push objects using the fitness function in (14).

[45] presents the evolution of a phototaxis and object avoidance behavior in a robot equipped with sonar and photosensors. A genetic programming structure implemented on an FPGA was used for the controller architecture. The behavioral fitness function used here is unusual in that it includes fitness values measured at previous time steps. The function is summarized as:

$$\varphi(t+1) = c_1\left(\varphi(t) + c_2(s_{\text{photo\_max}} - s_{\text{photo}}) + c_3\frac{\sum s_{\text{sonar}}}{s_{\text{sonar\_max}}} + c_4\right) \tag{15}$$

where s_{photo}, $s_{\text{photo\_max}}$, $\sum s_{\text{sonar}}$, $s_{\text{sonar\_max}}$ are the forward photo-detector excitation, the maximum photo-detector excitation, the sum of the sonar sensor excitations, and the maximum sonar excitations respectively. Notation note: the function is not in the form of an integrand (f) or an overall function (F), rather, it is presented in the original work as a recursive function and is here denoted by φ. Learning required 300 generations in addition to a 35-generation sensor tuning phase to develop functional controllers for the task.

[46] also evolved controllers for locomotion and object avoidance. The evolvable controller architecture was described in terms of artificial chemistries, a form of optimization algorithm based on the concepts of binding and reaction of compounds in chemistry. A Khepera with IR sensors was used. The evolvable controller architecture was a form of evolvable directed graph similar to a finite state machine. A behavioral fitness function with one term was used. The fitness function integrand minimizes the sum of differences between wheel speeds:

$$f_{(-)} = |v_l - v_r| \tag{16}$$

where v_l and v_r denote the right and left wheel motor speeds. Note that unlike most of the other locomotion and object avoidance experiments, no sensor activation term was used here, but the controllers still evolved successfully over the course of 160 generations. Controllers were evolved in simulation and the best resulting controllers were transferred to a real robot and tested in a small maze environment.

3.3. Functional incremental fitness functions

In this section we present ER research that used incremental fitness functions (summarized in Table 4). Recall that incremental fitness functions begin the evolutionary process in a form that selects for a simpler behavior than the final desired behavior, and then change their form to select for complex abilities. The function may change forms several times before the final task or behavior is achieved. To begin with, we will discuss early research from the mid-1990's in some detail (in particular the research performed at the University of Sussex [9]).

In [9] a differential goal homing task was investigated in which a robot must move toward a triangular target placed on a white wall while avoiding a rectangular target. The robot used in this work consisted of a camera mounted on an X–Y positioning gantry system. The gantry was placed over an arena, and evolution was performed with trials evaluated using the physical system. The work used a three phase functional incremental fitness function.

The first sub-function maximized the distance from the arena wall opposite the target by summing the robot's current distance from the wall (d_{wall}) at 20 time points over the course of a trial:

$$F_1 = \sum_{i=1}^{20} d_{\text{wall}}. \tag{17}$$

Here, we explicitly include the summation over 20 steps since it does not represent a true averaging of fitness and is trial-time dependent. F_1 might be considered a behavioral function. After fitness converged using F_1, the fitness function was replaced with F_2 and evolution was continued with a new population derived from the best-performing member of the first evolved population:

$$F_2 = \sum_{i=1}^{20} (-d_{\text{target}}). \tag{18}$$

F_2 is maximized when the distances d_{target} to the target (measured over the course of a trial) are minimized. F_2 might also be considered a behavioral function because it does not explicitly measure task completion. Note that the final form of an incremental fitness function can be classified as one of the non-incremental forms from the classification system of Section 2, but the intermediate forms are not necessarily classifiable unless they are intended to generate a specific behavior or task.

Likewise, a third fitness function was applied to a population derived from the best performing member of the previous population. Here, the single target was replaced with two targets, one triangular and one rectangular. F_3 is maximized when the distance from the triangular target is minimized and the distance from rectangle target is maximized (measured at 20 time points over the course of each trial):

$$F_3 = \sum_{i=1}^{20} \left(c_1(D_1 - d_{1i}) - c_2(D_2 - d_{2i})\right). \tag{19}$$

Here c_1 and c_2 are experimentally derived coefficients, D_1 and D_2 are the initial distances from the triangular and rectangular targets respectively, and d_{1i} and d_{2i} are the test point distances measured over the i time steps of each trial. Dynamic recurrent neural networks were evolved and were provided only two inputs from the camera. The areas within the camera's receptive field that led to activation of the two network inputs were also modified by the evolutionary process. Successful evolution of controllers capable of identifying and homing in on the correct target was reported after a total of 60 generations (20 generations using each of the three fitness functions). The three functions used might be considered together to be behavioral fitness functions because they do not measure task completion directly.

In [33] functional incremental evolution was applied to evolve a box-pushing and goal homing behavior for a Khepera robot. Genetic programming was used and controllers were encoded by tree representations that generated purely reactive controllers. A series of three incremental functions were applied to evolve the final behavior. Note that the authors minimized these functions during evolution rather than maximizing them. The fitness function integrands $f_{1(-)}$ and $f_{2(-)}$ were used to evolve the separate primitive behavior controllers of pushing a box in a straight line and box circumnavigation, respectively. $f_{1(-)}$ is given by

$$f_{1(-)} = c_1(1 - \text{mean}(v_l, v_r)) + c_2(|v_l - v_r|) + c_3(1 - s_{f\_ir}) \tag{20}$$

where v_l, v_r, c_1, c_2, and c_3 are used as in the previously presented standardized fitness function forms, and s_{f_ir} is the average current activation level of the two forward-most IR sensors on the robot. $f_{2(-)}$ is given by

$$f_{2(-)} = c_1(1 - \text{mean}(v_l, v_r)) + c_2(|s_{s\_ir} - c_3|) \tag{21}$$

where s_{s_ir} is the activation of a particular one of the IR sensors on one side of the robot that the designers chose to act as a distance regulator between the box and the robot. The constant values c_1, c_2, and c_3 selected in f_1 are different than those selected in f_2. Both functions can be classified as behavioral. A third controller evolution was performed to generate an arbitrator controller that was responsible for turning the primitive behaviors on and off, to produce the final goal homing behavior. The fitness function used was

$$F_3 = d_{\text{box,goal}} \tag{22}$$

where $d_{\text{box,goal}}$ measures the distance between the box and the goal location (indicated by a light source) at each time point during an evaluation trial. The final form of the function could be considered aggregate if the task were defined as getting the box as close to the goal as possible. It should be noted that only the primitive behaviors were tested in real robots, so it is not clear that the final evolutionary step was entirely successful.

The research reported in [71] used a two stage functional incremental fitness function to evolve locomotion and object avoidance abilities in a hexapod robot constructed using hobby servos. The robot was equipped with IR and photosensors. The overall behavior was evolved in two steps. First, legged locomotion was evolved using a fitness function of the form

$$F_1 = c_1 d + c_2 L \tag{23}$$

where d is the maximum distance achieved by the robot and L is a measure of activation to the leg actuators (the exact forms of the fitness functions used are not explicitly stated by the authors and the functions presented here were extrapolated from the text descriptions). The second stage of evolution generated object avoidance and made use of a simple single-term fitness function, F_2, for selection:

$$F_2 = d_{\text{collision}} \tag{24}$$

where $d_{\text{collision}}$ is the distance covered by the robot before it hits an obstacle during a given evaluation trial. Note that this is very close to being an aggregate fitness function but it does not measure full task completion directly. It makes the implicit assumption that avoiding collisions will result in the best locomotion. Controllers

were evolved in simulation and transferred to a real hexapod robot for testing. A neural network for locomotion was generated in the first phase of evolution, and then a second network was piggybacked on the first and evolved to generate the final collision avoidance behavior. It is not clear from this particular work how many generations were needed for each step of evolution.

In [36] populations of neural network-based controllers for Khepera robots were evolved to perform goal homing (phototaxis) and obstacle avoidance using incremental evolution in two stages. For this experiment, the robots used photosensors and IR sensors, but neural connections for the photosensors were not introduced until the second stage of evolution. Initially, controllers were evolved for straight-line motion with obstacle avoidance using the following fitness function integrand:

$$f_1 = c_1(v_l + v_r) - c_2(|v_l - v_r|). \tag{25}$$

The best controllers resulting from f_1 were then used as a seed population and were evolved further with f_2:

$$f_2 = c_1 s_{\text{front}} + c_2\,(|s_{\text{front}} - s_{\text{back}}|) \tag{26}$$

where s_{front} and s_{back} are the activation levels of the forward and backward photosensor arrays respectively. A population of 30 network controllers was evolved first for 100 generations with f_1 and then for an additional 100 generations with f_2. Both of these fitness functions are classified as behavioral.

The experiments discussed in [36] also contained an aspect of environmental incremental evolution. The experimenters placed additional obstacles in the environment and reduced the number of goal light sources over the course of evolution. This provided an environment of incrementally increasing difficulty. The best resulting controllers were tested in real robots and were demonstrated to be able to home in on a light source in environments containing obstacles (walls) arranged to force the robot to backtrack, and at times to explore dead ends in order to find the light source goal. [34] presents the evolution of a flight controller for beacon homing and circling. The controllers were evolved in simulation and then tested in a real ground robot that homed in on and circled a sonic beacon (EvBot II equipped with directional sound sensors). A combination of incremental and multi-objective selection was employed, using the following fitness function integrands:

$$f_{1(-)} = \frac{d}{d_0} \tag{27}$$

$$f_{2(-)} = B_{\text{inrange}} d^2 \tag{28}$$

$$f_3 = (1 - B_{\text{inrange}}) B_{\text{level}} \tag{29}$$

$$f_{4(-)} = B_{10^\circ\text{turn}}\,|\theta(t) - \theta(t-1)| \tag{30}$$

where d_0 is the initial position of the robot, d is the current position of the robot, and $\theta(t)$ gives the roll angle of the robot at time t. B_{inrange}, B_{level}, and $B_{10^\circ\text{turn}}$ are Boolean functions that are true when the robot is in range of the beacon, when the robot is level, and when the turn angle is greater than 10 deg respectively. Note that f_1, f_2, and f_4 are minimized while f_3 is maximized. Initially, f_1 was used to bootstrap evolution for 200 generations. After a level of homing confidence was gained, multi-objective optimization used all four fitness functions for 400 generations. In some ways, multi-objective optimization raises questions about the relationship between task definition and the definition of aggregate selection. If the task is explicitly defined in terms of solution features and the designer formulates these features into a set of multiple objectives, then this set of multiple objectives can be considered an aggregate selection mechanism when viewed as a whole. However, the choices made about which features of the solution should be jointly optimized, and the formulations of the functions for the objectives, requires a considerable amount of *a priori* task solution knowledge.

3.4. Tailored fitness functions

Tailored fitness functions contain aggregate terms that measure some level of task completion, but they may also contain behavioral terms. Aggregate functions that measure nothing but final task completion are classified separately and are presented in Section 3.8 of this section. Research using tailored fitness functions is summarized in Table 5.

In [47], Fuzzy logic rule sets were evolved to control a robot performing a goal homing and object avoidance task. A tailored fitness function with no behavioral terms was used.

The overall fitness function was stated in terms of two mutually exclusive cases, each with its own sub-function:

Case (1) A collision occurs:

$$F_{\text{collision}} = \frac{t_{\text{collision}}}{t_{\max}} \tag{31}$$

where $t_{\text{collision}}$ is the time at which the collision occurred and $t_{\max}$ is the maximum allowable number of time steps.

Case (2) No collision occurs:

$$F_{\text{free}} = \left(1 - \frac{d}{d_0}\right) \tag{32}$$

where d is the distance remaining between the goal and the robot, and d_0 is the initial distance between the robot and the goal at the beginning of the trial. A trial ended if a collision occurred, if the robot arrived at the goal location, or at $t_{\max}$.

After 30 generations controllers were evolved that were capable of reaching the goal location without collision in approximately 50% of the trials. This level of success is lower than that reported in other similar work [36,37], but at the same time, the evolutionary process was only allowed to continue for 30 generations. It is possible that controller populations had not yet converged.

In [48] the author evolves a simple locomotion with wall avoidance behavior in a robot using an evolvable hardware control system. The following tailored fitness function integrand was used:

$$f = \mathrm{e}^{-c_1(d_x)^2} + \mathrm{e}^{-c_2(d_y)^2} + B \tag{33}$$

where d_x and d_y are the distances from the center of the arena to the robot's current position, and B is a Boolean whose value is 1 if the robot is moving and 0 otherwise. The function is intended to keep the robot away from its starting position (the middle of an arena) and also keep it moving. This work was performed in a laboratory robot using sonar sensors. An unusual real robot and simulator evolutionary setup was used here. A real hardware controller using an FPGA was attached to a real robot during controller evaluation, but the robot was placed on a platform that did not allow its wheels to touch the floor. Wheel movements were measured and then fed into a simulator for final fitness evaluation. The motivation for doing this was to avoid simulating the entire dynamic controller-robot hardware system, but still allow the evolutionary process to proceed in a fully automated way. Functional controllers were evolved in 35 generations and tested in the real robot operating in a real environment.

In [28] an agent-herding task was investigated. A shepherd robot was evolved to herd a single agent with a fixed control strategy into a goal area. This work used two Nomad 200 robots, and along with [50] is one of the few evolutionary robotics experiments to use full-sized laboratory robots. The evolved control strategies consisted of a set of sensory stimulus-motor response rule sets. Controller rule sets were evolved in simulation and tested in the real robots. Fitness was measured in terms of percent, denoting partial or full success: 100% was given if the robot herded the sheep agent into the goal area before t reached $t_{\max}$. A constant c% was given if the robot herded the sheep agent

to within a predefined distance of the goal area when t reached $t_{\max}$ (with c selected to be less than 100 but greater than 0), and 0% was given in all other cases. This is essentially a tailored fitness function with two aggregate terms. In tests of the best evolved controller in the real robots, the herding robot was able to herd its companion robot into the goal area in 67% of test cases.

[14] reports on the evolution of a task in which a robot must pick up pegs in an arena and deposit them outside the arena. A Khepera robot with an attached gripper unit was used to test controllers evolved in a simulation environment. Sub-behaviors were identified and evolved (and one was hand-coded). A master coordination module was evolved to generate the final overall behavior. The method of fitness evaluation used here was extremely complex and must be considered more of a tailored and behavioral algorithm than a single function. Although in theory there could be many methods of performing the target task, the fitness selection algorithm allowed only one overall solution to evolve: the robot wanders through the arena environment until it detects a peg, it picks up the peg, it moves to the edge of the arena, and it drops the peg. The authors performed 10 separate evolutions of 1000 generations, using a population size of 100 individuals. They report the emergence of individuals capable of completing the task reliably (in over 90% of attempts) in simulation and in testing on the real robot. The peg collection and deposition task evolved here is among the most complex achieved to date. [35] investigated an almost identical task in simulation only, but used an aggregate success/failure fitness function in conjunction with environmental incremental evolution (discussed in Section 3.6 of this section).

[29] presents the evolution and testing of FPGA hardware controllers for a small robot with two cameras. The robot's task was to home in on a target object in an environment containing additional obstacles. The tailored fitness function required 64 full evaluation periods to be completed before application, and included an aggregate term that counted the number of outright completions of the task:

$$F_{64\_\text{trials}} = \sum_{1}^{64} \left(B_{\text{goal\_found}} + c_1 \left(1 - \frac{d_{\text{goal}}}{d_{\max}} \right) + c_2 \left(1 - \frac{t_{\text{goal}}}{t_{\max}} \right) \right) \tag{34}$$

where $B_{\text{goal\_found}}$ is a Boolean that is true if the robot found the target object during that trial, d_{goal} is the remaining distance between the robot and the goal at the end of the trial, t_{goal} is the number of time steps required to find the goal and $d_{\max}$ is the greatest travel distance (linear offset) that can be achieved in the training environment. Several variations on evolutionary conditions and methods were investigated, and the reader is referred to the paper [29] for a description of these. Using the most successful evolution strategy, the authors report that the best controllers from a population of 20 evolved for 650 generations and tested on a real robot were able to complete the task of finding the target object in a fairly complex environment containing many walls and starting from 64 separate initial positions within the environment.

In [49], controllers for Khepera robots were evolved for an object pushing and phototaxis task. In this task, robots locate pegs (small cylinders) within an environment and then push them to a goal location marked by a light source. For this experiment, the controllers were evolved in simulation and then transferred to real robots. The authors used the following fitness function:

$$F = c_1 \left(1 - \frac{d_f(\text{peg, goal})}{d_0(\text{peg, goal})} \right)^2 \tag{35}$$

where d_0 and d_f measure the distance between the peg and the goal light source at the beginning and the end of an evaluation trial. F falls into the class of tailored fitness functions because it measures some degree of success. The function does not inject a high degree of *a priori* knowledge into the controller strategies. In addition, as denoted by the upper case F, this is the full fitness function, not an integrand. In [49], populations of 100 standard feedforward networks and 100 partially Hebbian neural controller architectures were evolved in separate experiments for 200 generations. Following evolution, the qualities of the resulting controllers were compared. The best-evolved Hebbian controllers, which allow for some modulation of behavior during controller evaluation (often referred to as lifetime learning [87]), were more robust to noise in the robot's actuator system. The best Hebbian controllers were able to complete the task in 92% of trials (with actuator system noise) while the standard feedforward networks were only able to complete the task 75% of the time.

A controller for locomotion with object avoidance was evolved in simulation and transferred to an RWI B21 service robot equipped with sonar [50]. This robot is larger than most robots used in ER work (over a meter tall) and was tested in a building hallway, rather than in small specially constructed arenas used in other work. GP was used to evolve populations of controller programs and the authors formulated a complex fitness function for the relatively simple task. The tailored fitness function used here maximized time of motion while minimizing overall rotation of the robot over the course of a trial evaluation period. This can be written as:

$$F = 1 - \frac{t}{t_{\max}} \sqrt{\frac{\left|\sum r\right|}{\omega_{\text{net}} t_{\max}}} \tag{36}$$

where t is the time of the first collision, $t_{\max}$ is the maximum time allotted for an evaluation period, $\sum r$ is the sum of all discrete rotations made by the robot, and ω_{net} is the average angular velocity over the course of an evaluation period. One of the controller evolutions was performed on the real RWI B21 robot and required approximately 200 h of evolution time to perform 50 generations with a population size of 75. This highlights the need for high quality simulators if ER is to be used to generate controllers for robust robots operating in non-laboratory environments.

In [20] the authors report on the embodied evolution of a robot behavior in which a Khepera robot moves to an intermediate goal zone and then to a final home goal position. When the robot arrives at the intermediate goal position a light source is triggered. The authors used a simple tailored fitness function that injected limited amounts of *a priori* knowledge into the evolved solutions:

$$F = \frac{t_{\text{goal}}}{t_{\max}} \tag{37}$$

where t_{goal} is the number of time steps spent in the home goal position after the light has been triggered, and $t_{\max}$ is the total number of time steps per trial. As in [49], Hebbian neural networks were evolved for the task, and the learning rates of the networks were evolved, rather than the connection weights. Hence, the controllers were evolved to *learn* how to perform their task while in operation. Using a population size of 100 individuals, 500 generations were required to evolve competent networks. After evolution using Khepera robots, the fittest evolved networks were tested in Koala robots with similar sensor and actuator architectures and were found, as in [49], to be robust against changes in actuator response.

[52] describes experiments that evolve neural controllers for a box-pushing task. Here, the robot (a Khepera with IR sensors) must push an object toward a light source. The authors evolved separate populations of controllers using four different fitness

function integrands and then compared the quality of the evolved controllers. The four fitness functions are given by:

$$F = d_{\text{box}} - \frac{1}{2} d_{\text{box,robot}} \tag{38}$$

where d_{box} is the total distance that the box moved, and $d_{\text{box,robot}}$ is the final distance between the box and the robot;

$$f = \Delta d_{\text{box}} - \frac{1}{2} \left(\Delta d_{\text{box,robot}} \right) \tag{39}$$

where Δ (delta) indicates change over the current time step;

$$f = c_1 (s_{ir2} + s_{ir3}) + \left(1 - c_2 \sum s_{\text{photo}} \right) \tag{40}$$

where s_{ir2} and s_{ir3} are the forward facing IR proximity sensors and $\sum s_{\text{photo}}$ is the combined activation of 4 photosensors;

$$f = c_1 (s_{ir2} + s_{ir3}) + c_2 \left| v_l + v_r \right| - c_3 \left| v_l - v_r \right| \tag{41}$$

where v_l and v_r are the left and right wheel motor speeds.

All of these fitness functions converged upon a solution within 250 generations, using a controller population size of 30. The first and simplest function was reported as generating the best solutions. Only the controllers evolved with the first function were demonstrated in the real robot. Also, of the four fitness functions, the first contained the least *a priori* information about the task. This is interesting, and indicates that assumptions the researchers made about what features a good solution should have may not have actually been helpful in generating better solutions.

In [53], gaits were developed for a biped robot built from hobby servos. Embodied evolution was used to optimize a set of 12 gait parameters. The robot's vision and IR sensors generated information needed for fitness evaluation:

$$F = vD(\theta). \tag{42}$$

Here v is the average velocity of the robot over the trial period and $D(\theta)$ is a function that measures angular change in the robot's heading. D has a somewhat complex formulation but essentially rewards controllers that produce less rotation. F is used here to indicate that this is the complete fitness evaluation function, not an instantaneous integrand that is averaged or integrated over a trial period. This work started evolution with a working hand-formulated controller, and hence represents optimization rather than primary synthesis.

[54] investigated the evolution of three basic behaviors in the context of eventually learning more advanced behavioral coordination mechanisms. The behaviors were photo-orientation, object avoidance, and robot homing (come into proximity of another robot in the environment). Robots built mainly from LEGOs and equipped with IR, tactile and photosensors were used to implement a fully embodied evolution scheme. For each task, a simple tailored fitness function was used during evolution. The fitness functions are listed below:

$$F_{\text{phototaxis}} = \frac{\sum t_{\text{light}}}{t_{\max}} \tag{43}$$

where $\sum t_{\text{light}}$ is the number of time steps in which the robot was facing the light source, and $t_{\max}$ is the total number of time steps in the trial;

$$F_{\text{avoidance}} = \frac{\sum t_s}{t_{\max}} \tag{44}$$

where $\sum t_s$ is the number of time steps in which none of the IR sensors report activation; and

$$F_{\text{homing}} = t_{\max} - t_{\text{complete}} \tag{45}$$

where t_{complete} is the amount of time required to position the robot in proximity to another robot in the environment. The authors report successful evolution of behaviors within 30 generations for each of the three tasks. Unlike most of the other work reviewed, the genetic programming and controller structures used here contained some very high-level primitives, such as obstacle avoidance. It is unclear how much of the resulting control strategies were truly learned, and how much was encoded into the GP structure and learning environment.

In [12] a phototaxis task in which robots learn to home in on a light source was investigated. Evolution was performed in a population of eight real robots using an asynchronous algorithm and a tailored fitness function. By asynchronous we mean that the evolutionary algorithm did not employ any specific generation or epoch period. Rather, a controller might propagate as soon as it had achieved a high enough fitness level, regardless of what the other controllers were doing. During evolution, fitness was considered to be the current *energy level* of each robot and increased each time the robot reached the light source, and decreased during reproduction when a robot would broadcast its genes to other population members. The fitness function φ calculates the energy level and is updated at each time step. It can be summarized as follows:

$$\varphi(t) = \varphi(t-1) + c_1 B_{\text{reward}} - c_2 B_{\text{penalty}} \tag{46}$$

where t is time, B_{reward} is a Boolean that is true in any time step when the robot reaches the light source, and B_{penalty} is a Boolean that is true in any time step where the robot broadcasts its genes. The fitness is limited to a maximum value. The algorithm is fully asynchronous and there is no population-wide trigger for reproduction. Reproduction (broadcasting and receiving of genes) is governed probabilistically based on robot energy (fitness) levels. After about 100 min of evolution, fitness improvement in the population leveled off. Controllers capable of approaching the light source using a variety of strategies were reported.

In [56] embodied evolution was used to generate neural network controllers for locomotion and wall avoidance in a Koala robot equipped with a vision system. The authors used a simple fitness function integrand that maximized the forward velocities of the wheel motors:

$$f = (v_l + v_r) - (|v_l - v_r|) \tag{47}$$

where again v_l and v_r are left and right drive motor speeds. Fitness was integrated over the time steps of a trial, and then integrated again over a given number of trial periods. Note that unlike the works of [11,13], no explicit sensor activation term was used. This allowed the evolutionary process more freedom to evolve novel solutions. The underlying task investigated here is quite simple, and has been studied in many previous research efforts. In most previous work, though, IR and photosensors were used. Here, a grey-scale image from the vision system was partitioned into a 5 by 5 grid, and the average light level of each grid cell was used as a network input. Using a population of 40 neural controllers, fit controllers able to navigate around the simple environment were evolved within the physical environment in only 15 generations.

[30] discusses the evolution of a coordinated movement task in which three robots must move together in formation. The robots used a typical differential drive system and each had four IR sensors for detection of the environment. The authors use a relatively complex tailored fitness function given by

$$F = P \sum_{1}^{T\max} \left(D_{\text{gain}}(d, d_{\text{best}}) \left(1 + \tanh\left(\frac{S}{20} \right) \right) \right) \tag{48}$$

where P is a collision penalty term that decreases toward 0 as the number of collisions increases, D_{gain} is a function of present

distance d and trial-best distance d_{best}, and S is a measure of team dispersion. In this case we included the summation term explicitly in the fitness function representation because the collision penalty factor P is applied after the integration over the evaluation time period of the other elements of fitness evaluation. The function is represented by an uppercase F, in accordance with the nomenclature used in this paper. The function is relatively selective for a class of *a priori* known solutions, but is not explicitly selective for an exact known solution. In [30] the authors used a population of 50 controllers and ran 100 separate evolutions. They report that in every evolutionary run, a fit controller eventually arose that was capable of achieving the group locomotion task in simulation, and also when tested in the three real robots.

[57] studied the evolution of locomotion and object avoidance behaviors using evolvable hardware controllers (FPGA). Embodied evolution was used and fitness evaluation was performed in a physical Khepera robot with FPGA turret. The following fitness function was used:

$$F = c\frac{d(1-\sum s)}{\sum rev} \tag{49}$$

where d is the distance traveled by the robot, $\sum s$ is the sum of all sensor activations, and $\sum rev$ counts the number of discrete motor direction reversals. This fitness function is related to the behavioral functions used for locomotion and object avoidance in earlier works [11,13,56], but here, distance traveled d (an aggregate term) is used, rather than wheel motor speeds. The authors also include the unusual behavioral term $\sum rev$ counting the number of motor reversals over the course of a trial period. Successful controllers were evolved within about 20 generations.

As part of a larger layered control architecture, in [58] the authors evolved fuzzy logic controller modules for object (ball) homing behaviors in a Sony AIBO. Here an evolvable fuzzy logic controller architecture was evolved for control. A tailored fitness function with three aggregate terms of the following form was used:

$$F = (1-c_1 d)(1-c_2 a_{\text{goal}})(1-c_3 t_{\text{goal}}) \tag{50}$$

where d is the final distance between the target object (a ball) position, a_{goal} is the final angle between the target object position and the robot's head, and t_{goal} is the amount of time elapsed during the trial. c_1, c_2, and c_3 are chosen to normalize the various factors. Note that this fitness function implicitly includes a stopping condition when the target is found. Unlike many other ER experiments, this particular work included a fair amount of hand-coded control elements, including sensor fusion and object identification, and predefined gaits. In the fuzzy logic controllers, the antecedents of the rules were predefined, while the consequences of the rules were evolved, and this may have introduced a high level of additional *a priori* task knowledge into the resulting evolved controllers. The task of object homing was evolved in the real robot in 20 generations using a population of only 10 individuals.

Neural network controllers for object avoidance and locomotion were evolved using embodied evolution in a system of six real robots and also in a simulated version of the system [55]. The robots were custom-built differential drive systems with IR and tactile sensors. Fitness over a given trial period was calculated using the following set of rules: (1) Start with 4096 points; (2) Receive 10 points for each second of forward movement; (3) Lose 30 points for any occurrence of a forward motion command that is shorter than 15 s in duration; (4) Lose 10 points for each collision that occurs in conjunction with a forward motion command. In terms of selection, this set of rules produces an effect similar to some of the other behavioral and tailored fitness functions but it is unusual in that it includes actual motor commands explicitly. Element 2 makes the function tailored since it measures a degree of task completion. Elements 3 and 4 are behavioral terms. As in the case of [12] the evolutionary algorithm studied in [55] was intended to operate within the six physical robots. During each generation, the fittest robot controller was transferred to the other five robots where it was combined with the local controller using crossover. The authors explored various mutation rates, including a form of periodic macro-mutation couched in terms of predation. Robot controllers able to maximize the fitness function arose during the course of 200 generations in experiments run in the real robots.

[59] discusses the evolution of locomotion and object avoidance behaviors using an EvBot robot equipped with 5 binary tactile sensors. The EvBot robots [96] are small cylindrical robots between 15 and 25 cm in diameter, and can be equipped with a variety of sensors. These robots have more computing power than typical robots of this size and generally run full PC operating systems as well as high-level computing packages on board. Neural controllers were evolved to perform the navigation task, and a tailored fitness function was used:

$$F = c_1 d_{\text{net}} + c_2 d_{\max} + c_3 d_{\text{arc\_length}} - c_4 B_{\text{stuck}} \tag{51}$$

where d_{net} measures the offset distance between the robot's starting and its final position, $d_{\max}$ measures the greatest distance achieved by the robot at any time during the trial, $d_{\text{arc\_length}}$ is the line integral arc length of the robot's path over the course of a trial and B_{stuck} is a Boolean that is true if the robot becomes permanently immobilized during the trial. Because only five binary sensors were used to detect the environment, the robot's perceptual space contained only 32 distinct states, hence a simple reactive controller would be sub-optimal. To compensate for this, recurrent neural networks with several hidden layers and capable of temporal signal processing were used. The authors report that effective controllers were evolved in simulation and tested on real robots using a population size of 20 controllers and required on the order of 3000 generations. This number of generations is quite high compared to much of the other ER work surveyed, but at the same time, most other work resulted in the evolution of simple controllers that were purely reactive. The controllers in [59] evolved time-memory control solutions that compensated for the extreme temporal aliasing introduced by the binary tactical sensor system used.

Gaits for a biped servo-bot were evolved in simulation and then demonstrated on a real biped robot in [60]. Evolvable spline controllers were used in this work. Splines controlling each joint actuator were coordinated based on gait cycle time, and the evolutionary process altered each joint actuator spline's defining control point parameters. During evolution, a tailored fitness function that measured how far the robot moved was used:

$$F = c_1 d - c_2 v_{\text{body\_lowering}} \tag{52}$$

where d is the distance traveled by the robot and $v_{\text{body\_lowering}}$ is the average downward motion of the robot's torso. This term is included to reduce the amount of body movement of the robot to create a smoother gait. This fitness function is classified as tailored because it measures a degree of success of task completion, i.e. how far the robot moved, regardless of how that movement was achieved. The second term of the function is a behavioral term that selects for gaits with minimal movement in the torso. As with some, but not all, of the research into gait learning, in this work no sensors were used and the robots did not learn to dynamically interact with their environment. The evolved controllers were able to generate balancing and walking both in simulated and real biped robots.

Gaits for a Sony AIBO robot were evolved using embodied evolution in [61]. A gait parameter set was evolved. The fitness function used is very similar to the one used in [53] and derived all

required information from images received from the robot's head-mounted camera:

$$F = vD(\theta). \tag{53}$$

Here v is the velocity of the robot over the trial period and $D\theta$) is a function that measures change in the robot's heading. D rewards controllers that produce less rotation. Robots were reported to travel at speeds of 1 m/min using the best evolved set of gait parameters. Although images from the robot's camera were used during evolution to determine fitness, the robot did not learn to react to an object or other elements of its environment based on sensor information. As in many of the other gait-learning experiments, the learned control was not dynamic with respect to the environment. The evolutions required between 300 and 600 generations using a population of size 30. This number of generations is quite high compared to other embodied work.

In [62], controllers for locating and pushing objects toward a goal region were evolved in simulation and then optimized in real robots. A form of Q-learning was used, in addition to a genetic programming phase, for a hybrid system that cannot be purely labeled as evolutionary robotics. In order to accommodate the Q-learning phase of control learning, a state space was formulated based on a classification of images from the robot's camera. Further, the robot's possible actuator command set was similarly formulated into a finite set of states. A complicated tailored fitness function was used and is summarized by:

$$F = c_1 B_{\text{goal}} + c_2\left(1 - \frac{moves}{moves_{\max}}\right) + c_3\left(1 - \frac{turns}{turns_{\max}}\right) + B_{\text{box\_moved}} + B_{\text{goal\_seen}} - \frac{goal\_lost}{t_{\max}} \tag{54}$$

where B_{goal} is a Boolean function that is true if the robot moves the object to the goal area before the trial period is over, *moves* is the number of linear moves made by the robot, *turns* is the number of turning moves made by the robot, $B_{\text{box\_moved}}$ is a Boolean function that is true if the robot manages to move the object at all, $B_{\text{goal\_seen}}$ is true if the robot positively detects the goal region at any time during the trial, and *goal_lost* counts the number of times the robot turns away from the goal region after it has first detected it. The terms $\text{moves}_{\max}$, $\text{turns}_{\max}$ and $t_{\max}$ refer to maximum allowable numbers of moves, turns and time steps per trial period respectively. These three terms were set by the researchers based on their expertise and understanding of the experimental setup. In [62], the authors reported that successful controllers were generated in simulation and then tested and automatically optimized in real robots.

A relatively difficult sequential task in which a robot must visit three goal locations in a specific order is investigated in [22]. Populations of small recurrent neural networks (6–10 neurons) were evolved with an extended multi-population genetic algorithm. Most other ER researchers use single population genetic algorithms, and the work in [22] compared the two forms of evolution and reported superior results using multi-population evolution. During a given trial period, fitness is updated by:

$$\varphi_{\text{goal\_position}} = \begin{cases} 1 & \text{if in sequence} \\ -1 & \text{otherwise.} \end{cases} \tag{55}$$

Here fitness is updated only when the robot reaches any one of the goal locations. An additional single overriding behavioral condition is also included: a robot that produced only spinning-in-place behaviors is given -30 points. The main fitness function contains relatively little *a priori* task solution knowledge and uses only information related to the completion of the cycle of goal visitations. Successful controllers were evolved after 50 generations using populations of on the order of 200 individuals. This is well within the range of generations required for evolution reported by other researchers for other less complex tasks.

Controllers capable of performing a phototaxis task in an environment with many occluding obstacles were evolved in [63]. A robot constructed from LEGO Mindstorm kits and equipped with two photosensors and a single forward-mounted tactile sensor was used. A simple tailored fitness function was employed, and is given by:

$$F = d_{\max}^2 - d^2 \tag{56}$$

where $d_{\max}$ is the largest dimension of the training environment, and d is the final distance between the robot and the light source after an evaluation trial period. The authors report successful evolution of controller programs after 350 generations and using a population size of 64. Control programs were evolved in simulation and then tested in the real robot. The best evolved control program was reported to be able to complete its task (finding a light source) in all 15 trials in the real robot.

In recent work [110] neural network-based controllers were evolved to perform a coordinated group movement task in which interconnected robots travel together in an environment containing holes. An aspect of robot–robot communication was involved in this work, and robots were given the ability to produce and receive signals in the form of a tone. To perform the task, the robots (arranged in a square) must move in as straight a line as possible while maintaining formation and avoiding holes. Evolution of controllers was performed in simulation and then evolved controllers were tested in physical robots. A complex tailored fitness function that combined individual robot fitnesses and success at hole avoidance was used. The individual robot fitness was measured using a function similar to Eq. (3) but with additional tailored factors. This function included elements that selected for rapid movement using a normalized version of the first factor of Eq. (3), straight movement using the second factor of Eq. (3) with a zero-floor condition, as well as a factor that measured the degree of coordinated traction force produced by the robots. The function contained additional factors that measured the degree of ground sensor activation (used to detect the holes) and a term intended to minimize robot–robot communication. The fitness of the robot group was 0 if any member of the group fell into a hole, and a function of the lowest individual robot fitness otherwise. In addition to the explicit fitness function used in this work, three separate environments, two of which did not contain holes were used during evolution. Although each fitness evaluation trial combined results from all three environments, in some respects, this represents a form of environmental incremental evolution (discussed in Section 3.5) since robots could realize fitness gains early in evolution by improving simple locomotion abilities in the environments without holes.

3.5. Environmental incremental fitness functions

Environmental incremental evolution (research listed in Table 6) differs from functional incremental evolution in that the difficulty of the environment is augmented, and not the fitness function. Potentially, this can produce controllers evolved with aggregate selection and less explicit human bias. Some degree of human bias is still injected into the evolving controllers due to the selection of the various incrementally more challenging environments.

In [37] a goal homing and object avoidance behavior was evolved using a robot equipped with IR sensors and binary photosensors. Neural network controllers for the robot were evolved in two sequentially more difficult environments. The first environment included a single simple wall-like obstacle, while the second contained a concave cul-de-sac obstacle that required the

robot to learn to initially backtrack away from the goal in order to eventually approach it. The fitness function integrand used in both environments takes the following form:

$$f = \left(D_{\text{goal}}\right) \vee_{\text{xor}} \left(|\text{mean}(v_l, v_r)|\,(1 - \sqrt{|v_l - v_r|})(1 - s_{ir})\right) \quad (57)$$

where v_l and v_r are left and right drive motor speeds, and s_{ir} is the greatest current activation level of the IR sensors. D_{goal} is a measure of proximity to the goal. Note that the two parts of the fitness function are mutually exclusive; only one can apply at a time. 200 generations were performed in the simpler environments, followed by 200 additional generations in the more complex environment. Populations of 100 network-based controllers were used in this work, and the authors compared recurrent and non-recurrent neural network architectures in separate experiments. The recurrent networks were able to achieve higher levels of performance (they reached the goals more quickly), but the non-recurrent networks were still able to learn (i.e. evolve) to perform the task.

Environmental incremental evolution is also studied in [35]. The authors investigated a fairly complex object acquisition and deposition task in which a simulated robot equipped with a gripper must find and pick up a peg in an arena and then deposit it outside the border of the arena. A single fitness function was used while the conditions of the environment were changed incrementally to produce an increasingly difficult task. The fitness function used is given by

$$F = t_{\text{max}} - t_{\text{finish}} \quad (58)$$

where t_{max} is the maximum allowable time per evaluation trial, and t_{finish} is the time at which the task is completed. Note the function F in (58) provides the final fitness evaluation for a given trial and is not an integrand. Controllers were evolved in three stages, each with an incrementally more difficult environment. These were: (1) In the first stage, robots began each trial already holding the object, so the task would be completed when the robot had moved to the edge of the arena and dropped the object; (2) In the second stage, robots began each trial with the object directly in front of their grippers; (3) In the final stage, the robots began each trial at a random position within the arena.

Note that F in (58) can be classified as an aggregate fitness function and injects no *a priori* information into the evolved solution. However, the selection of incrementally more difficult training environments does restrict the evolved solution to a degree, and the selection of these training environments required knowledge of a feasible solution by the designers. The three environments listed above were used for 100, 400, and 100 generations respectively, and the best evolved neural controllers were able to perform the overall task when tested in the simulated robot.

3.6. Competitive fitness evaluation

Competitive and co-competitive evolution in which the fitness of one individual may directly affect the fitness evaluation of another individual represents an important but relatively small subset of the research surveyed in this paper. Several examples of co-competitive evolution, in which two distinct populations compete against each other asymmetrically (e.g. predator and prey robots), have been reported in the literature (research listed in Table 7).

The research presented in [10] co-evolved pursuit and evasion behaviors in differently configured species of predator and prey robots. The authors used competitive aggregate fitness selection methods. The fitness functions for the competing robot species were based on the time at which contact between predator and prey occurred and can be summarized as follows:

$$F_{\text{prey}} = \frac{t}{t_{\text{max}}} \quad (59)$$

$$F_{\text{predator}} = 1 - \frac{t}{t_{\text{max}}} \quad (60)$$

where t is the time at which contact occurred and t_{max} is the maximum length of time allowed for the evaluation trial periods. These paired fitness functions are considered aggregate because they involve only information pertaining to completion of the task, and not information related to low-level behaviors. As is often the case with aggregate fitness functions, F_{prey} and F_{predator} generate weak (but unbiased) fitness signals. To compensate for this, fitness for the evolving individuals was averaged over several complete sets of trials before selection occurred. Two populations of 100 controllers each were evolved (one for the prey and one for the predator) for 100 generations. The resulting populations of controllers achieve an initial level of competence early in evolution (after 25 generations), and then begin to cycle through reciprocal levels of higher and lower performance. The authors show that this performance cycling can be lessened to a degree by using hall-of-fame selection rather than a simple greedy selection method.

Competitive evolution of neural network-based controllers was investigated in [21] using EvBot robots equipped with color vision systems. Teams of robots competed against each other to find separate goal objects placed within a complex maze environment. The work was described in terms of a competitive game in which each robot team must try locate their opponent's goal or home marker object before the other team can locate theirs. During each generation, a tournament of games was played between teams of robots, in which the robots on one team would be controlled by copies of one controller network from the evolving population and the robots on the other team would be controlled by copies of another network from the same population. If any single controller in the population was able to win a game, then all controllers in the entire population were evaluated using the following aggregate fitness function:

$$F = 1.5wins - .5draws - 1losses \quad (61)$$

where *wins* is the number of wins achieved by the controller during a tournament, *draws* is the number of games played to a draw, and *losses* is the number of games lost by the controller. If no single controller within the entire population could win a game, then fitness selection reverted to a tailored bootstrap selection mode summarized by:

$$F = d_{\text{max}} - c_1 B_{\text{stuck}} - c_2 B_{\text{motor}} \quad (62)$$

where d_{max} is the maximum distance traveled by the robot, B_{stuck} is a Boolean that is true if all robots on a team become immobilized, and B_{motor} is a Boolean that is true if robots on a team generate motor commands that exceed the capabilities of the drive motors. Populations of 40 controllers were evolved for 450 generations. The best evolved controllers were tested in real robots and shown to be able to compete with hand-designed controllers created to play the same competitive game.

3.7. Aggregate fitness functions

Unlike behavioral fitness functions, which measure only aspects of a robot's behavior during testing, and tailored fitness functions, which may contain behavioral terms, aggregate fitness functions measure only task completion divorced from any specific sensor-actuator behaviors. Aggregate fitness functions collect (or aggregate) the benefit (or deficit) of all aspects of a robot

controller's expressed abilities into a single term. The fitness of an evolving controller is calculated based only on whether or not it completes the task it is being evolved to perform. If the task can be completed in a well-defined way, the fitness function will use only success/failure information. For example, if the task involves competing in a win-lose game, the fitness function will include a Boolean whose value depends only on whether the game was won or lost in a particular trial period. If the task can be measured by a final achieved quantity, then it will consist of a single scalar term. For instance, in a foraging task, an aggregate fitness function might simply count the number of objects correctly collected at the end of a trial period. Research using aggregate fitness functions is summarized in Table 8.

Using aggregate fitness functions, controllers have been evolved for tasks including gait evolution in legged robots [64,32,69,70], flying lift generation in a flying robot [66], and simpler locomotion [16,65,67,68] and object pushing [17] tasks. The simpler actuator coordination tasks are less environmentally situated and produce less complex reactions to environmental stimuli. In some cases, the robots do not have sensors at all. However, these works are included here for their application of evolutionary computation to design novel controllers.

In [17] the evolution of a ball-pushing behavior using a Sony AIBO is described. The function measures the degree of success of moving the ball simply by measuring the linear distance between the ball's starting and ending locations:

$$F = d_{ball}. \tag{63}$$

Here, fitness for each individual was averaged over several evaluation trials before propagation of the population to the next generation took place. This reflects an attempt to boost the limited fitness signal generated by this aggregate fitness function. In this work, information from vision and IR sensors was fused to generate virtual sensor information including angle to ball and apparent size of ball. Populations of 60 neural network controllers were evolved in simulation for 100 generations. The fittest controllers were then demonstrated in a real robot.

In [64] the authors use embodied evolution to develop gaits for a hexapod robot. The gait controllers were in the form of evolvable state lookup tables. An aggregate fitness function was used that measured the distance traveled by the robot while walking on a treadmill:

$$F = d. \tag{64}$$

The researchers reported evolution of functional gaits after 23 generations in a real robot using a controller population of size 30.

Both [16,65] describe separate examples of systems in which whole robots (bodies and controllers) were co-evolved in simulation and then constructed in the real world using modular actuators and structural units. In both cases robots were evolved for locomotion abilities and fitness was calculated simply as the distance d traveled. This was a completely aggregate fitness function and contained no other features of potential control solutions or of possible robot morphologies:

$$F = d. \tag{65}$$

These two separate research efforts evolved agents capable of locomotion, but without any sensors, and thus the evolved control structures did not interact dynamically with their environments to actively avoid obstacles or perform other tasks that might be expected of an autonomous robot. However, both systems produced functional designs that were used to construct real robots that were tested and found to be able to move in the real world.

[32] reported on the embodied evolution of a locomotion behavior in which a robot (built from a LEGO Mindstorm kit and relying on tactile sensors for object detection) must travel around an environment containing a single circuit and small obstacles placed along the walls. The controller architecture used here was a simple mapping from sensor activation states to motor states. Although not referred to as such in the paper, this was in essence a simple neural network with linear excitation functions and constant weights. The fitness function simply measured the distance traveled by the robot (as reported by a swivel wheel and odometer attached to the robot) over a given evaluation period:

$$F = d_{\text{arc\_length}} \tag{66}$$

where $d_{\text{arc\_length}}$ is the line integral (arc length) of the path traveled by the robot, rather than the net displacement. This fitness function can be considered aggregate if the task is considered to be simple locomotion in an unknown environment. After 20 generations, the best evolved controller was able to produce a continued locomotion speed of 3.5 m/s without colliding with obstacles. This compares to a slightly slower speed achieved by a hand-coded controller designed to perform the same task.

In [66] embodied evolution is used to develop lift-generating motions in a winged flying robot. A simple fitness function was used that measured height obtained by the robot at each time step:

$$f = h. \tag{67}$$

A simple genetic programming-based controller representation capable of expressing wing angles, positions and time durations was used. A very small population of 4 individuals was evolved with a tournament selection genetic algorithm. The robot did learn to generate lift, but the robot was unable to generate sufficient lift to completely loft the robot under its own power. Hence, the evolution process was not entirely successful.

In [67] an indoor floating robotic blimp equipped with a camera and placed in a small room with bar-code-like markings on the walls was evolved to produce motion and wall avoidance. Populations of neural network-based controllers were evolved. A fitness function was used that averaged magnitude of velocity over each trial period:

$$f = v \tag{68}$$

where v is the current velocity of the robotic blimp at each time step. The function is considered aggregate when selecting for the task of movement. The aspect of object avoidance is included only implicitly. Using a population size of 60, successful controllers were evolved in 20 generations with evaluations performed on the physical robot.

[68] described the evolution of morphology and recurrent neural network-based control for sensorless modular robots constructed of LEGO and servo units. Robots were evolved for locomotion abilities in simulation and then constructed in the lab with real hardware. An aggregate fitness function (the same as that used in [16,65]) was used that measured total net locomotion distance d over the course of a trial period:

$$F = d. \tag{69}$$

Note that an initial settling period occurred before each fitness-measuring period began. This was done to avoid selecting for robots that moved merely by falling and this makes the fitness function technically tailored, to a small degree. Evolution of functional locomotion abilities required on the order of 2000 generations. The best evolved robot and controller was constructed and was reported to be able to move approximately 14 cm per minute. In its virtual evolution environment, the simulated version was able to move about twice this distance in the same number of controller update cycles.

[69] presents another example of the embodied evolution of gaits in a physical robot. The robot was a pneumatic hexapod of

minimalist design. The authors used the same aggregate fitness function as did [16,64,65,68]. Distance for the aggregate fitness function was determined using images taken from an overhead camera. The evolvable controller structure consisted of a set of gait parameters within a control program looping structure.

Gait learning using a slightly different aggregate fitness function is given in [70]. In this paper, a Sony AIBO quadruped robot was used, and the evolutionary process was embodied in the real robot. Evolved controllers took the form of a set of 12 gait parameters. Fitness was measured as average speed achieved by the robot:

$$F = \frac{d}{t_{\max}} \tag{70}$$

where $t_{\max}$ is the time length of a given evaluation trial. Note that since $t_{\max}$ is constant, this function reduced to that used in [64,69]. The gaits evolved in [70] were reported to allow the robots to travel 20% faster than the best gaits resulting from hand-tuned parameter sets.

3.8. Research using simulated robots

In preceding sections of this survey we have concentrated on work that involved real robots in one form or another. ER work requires some level of physical verification. Research done with agents in a purely simulated environment with no association to any particular physical system is often classified as artificial life research. Even so, there is a quite large body of work billed as evolutionary robotics research that uses only simulated robots, or animats [103,104]. Simulation plays a major role in evolutionary robotics, and the case can be made that simulation-only work can be as valid as work verified in real robots, if the simulations are designed carefully. The work done in the 1990's demonstrated the validity of many simulated systems by directly verifying results in real robots. At the same time artificial life research and some work billed as evolutionary robotics continues to make use of simulated sensors and actuators that incorporate global environmental knowledge, sometimes in very subtle ways. The great majority of the ER research reviewed above did involve real robots, primarily in the verification phases. In only a very few cases, simulation-only work was listed because it was involved with a project that used real robots or because it was the only example of work of its type.

In this subsection we briefly list some of the major simulation-only ER research reports, starting with those that used behavioral or tailored fitness functions.

[72] reports on a cellular encoding scheme for evolvable modular neural networks for simulated legged robot control. An example of a relatively complex task achieved in simulation using a tailored fitness function is presented in [73]. The authors describe the evolution of a coordinated movement task involving several simulated robots. In [74] the authors studied the evolution of simulated robot controllers for a task in which a robot must collide with objects ("collect" them) in one zone and avoid them in another. Another example of evolving simulated robot controllers to perform a (relatively) complex task is reported in [75]. There, robot controllers evolve to produce lifetime learning in order to predict the location of a goal object based on the position of a light source. [76] presents experiments to evolve a group-robot flocking behavior in simulated robots. A simulated two-robot coordination task in which two robots evolve to move while maintaining mutual proximity is reported in [77]. In addition, the research in [78] evolved homogeneous controllers for a task in which four simulated robots must move together in a small group toward a light or sound source. In [79] groups of simulated robots (the Swarm-bots) evolve group attachment and aggregation abilities as well as group locomotion and inverted obstacle (hole) avoidance abilities. These robots have since been built [80], and [110] reports recent tests of the evolved controllers in the real robots. Other examples of behavioral and tailored fitness functions used for the evolution of behaviors in simulated robots are found in [81].

In [82] controllers created using incremental evolution in simulation are studied. In [83] the authors study functional and environmental incremental evolution and multi-objective optimization in a simulated aerial vehicle. Further application of multi-objective optimization applied in simulated evolutionary robotics systems is found in [8] as well as an extensive review of related work. [84,39] investigated the simulated co-competitive evolution of competing populations in the form of predator–prey behaviors. Finally, the co-evolution of controllers and morphologies is studied in simulation in several works including [85,51,38].

4. Discussion

The literature contains a large amount of experimental repetition, at least in terms of tasks studied and fitness functions used. Looking through the entire body of evolutionary robotics work involving real robots, we find that there are only a handful of distinct tasks for which controllers have been successfully evolved. The most common among these are: (1) simple locomotion and basic actuator control for locomotion [16,42,53,56,60,61,64–70]; (2) locomotion with object avoidance [48,13,11,26,27,15,32,43,46,50,55,57,59,71,110]; (3) goal or position homing [12,34,58]; and (4) goal homing with object avoidance [29,36,37,45,47,63]. These tasks can be considered as a set of benchmark experiments for the field of ER.

In some ways, a set of *de facto* benchmark tasks does not benefit the field of ER as much as would be the case in many other fields. The emphasis in much of the current ER research is on evolving more complex behaviors [14,20–22], and ultimately more general behaviors.

4.1. Fitness assessment methods and novel task learning

In the larger field of evolutionary computing, honing of methods and optimization of algorithms play a central role. Many of the researchers who conduct experiments in the field of ER, and whose work has been discussed in this survey, also address issues of algorithm efficiency and evolutionary conditions [5,29,43,55]. Reducing the time needed for evolution [5] and minimizing controller size [43] are examples of particular aspects of evolutionary system efficiency that have been addressed in ER experiments. However, because ER is largely focused on developing novel behaviors not seen in earlier work, and on increasing the complexity of evolvable behaviors, efficiency issues take a back seat to the more fundamental issue of fitness assessment. It is necessary to use controller representations that do not significantly restrict the controller search space and that do not result in intractable evolutionary conditions, but beyond this, efficiency is not the limiting factor in the current state of the art of ER. The fitness function governs: (1) the functional properties of the evolved controllers; (2) the point at which training plateaus due to lack of fitness signal; and (3) the degree to which novel behavior is learned. Robot controllers capable of performing new, more complex, tasks are evolved when researchers devise new fitness functions capable of selecting for the particular tasks of interest. Current state of the art ER research employs fitness functions that can select for relatively simple controllers capable of performing tasks composed of no more than three or four coordinated components. (Tasks investigated are summarized in the introduction to this section.)

One might then inquire as to which classes of fitness functions are most effective at selecting for more complex behaviors. Some work has been done comparing the effectiveness of different fitness functions aimed at evolving controllers for the same task. [52] investigated four different fitness functions applied to a box pushing task and found that all of the fitness functions produced reasonably competent controllers. In addition, research in [21] compared environmental incremental evolution to standard single-environment evolution and found no clear advantage to either form of selection. Neither of these research efforts produced truly statistically significant results that might be generally applicable to a wider range of ER applications.

In general the relative fitnesses or qualities of the evolved controllers in different research efforts are not known. Because of this, results achieved in different research efforts using different fitness functions are difficult to compare in absolute terms. Even so, some comparison can be made. All else being equal, if two ER platforms produce robot controllers capable of performing similar tasks, with one platform using a complex hand-formulated fitness function and the other using an aggregate fitness function, then one can say that the platform using the aggregate fitness function is extracting more information from the environment during evolution. From this point of view aggregate fitness functions generate a greater degree of novel environment-based learning.

4.2. Can aggregate selection generate complex behavior?

In the early 1990's the first successful ER experiments generally employed behavioral and tailored fitness functions. Since that time, many varied forms of fitness functions have been applied by different researchers to evolve controllers.

For each of the common tasks that have become benchmarks in ER, there are examples of aggregate fitness functions or fitness function contains relatively little *a priori* task solution knowledge that have been applied to successfully evolve functional controllers for real robots [16,65,32,68,63]. The majority of such work has been accomplished since the year 2000 and represents a shift informed by experiment within the field of ER [107].

The existence of examples of aggregate selection being used to drive the evolution of controllers for a variety of tasks indicates that many of the arguments presented in justification of the various more complex fitness functions are not in fact completely sound, at least for these simple tasks. Some of the most complex behaviors evolved did require very complex fitness functions [14,110], but interestingly, several of these most complex evolved behaviors were also evolved in other experiments using fitness functions containing relatively little *a priori* task solution information [20,35,21,22].

Aggregate selection bases selection only on success or failure to complete the task for which controllers are being evolved. The other forms of fitness functions currently used in ER require designers to understand important features of the tasks for which robots controllers are being evolved. In extreme cases, the fitness function essentially defines an *a priori* known solution. If aggregate selection could be achieved for much more complex tasks, it could eventually lead to the application of ER methods to environments and tasks in which humans lack sufficient knowledge to derive adequate controllers.

In order to apply aggregate fitness selection methods, the *bootstrap problem*, in which randomly initialized populations have no detectable level of fitness, must be addressed. Sub-minimally competent initial populations cannot be evolved using only aggregate fitness functions. There are several methods that might be used to overcome this difficulty and still allow for the use of aggregate fitness selection. Some such methods have been applied in research reviewed in this paper. These include: (1) applying environmental incremental evolution in conjunction with aggregate selection [35]; (2) using a bootstrap mode that gives way to aggregate selection later in evolution; and (3) applying competitive evolution so as to create an environment that continually increases in difficulty due to the evolving skills of other controllers in the population or agents in that same environment [21].

Although it may be possible to overcome some of the problems with simple aggregate selection, ER research still has not generated controllers capable of performing truly complex tasks. The great majority of tasks investigated in current and past ER research are simple enough so that un-modified aggregate selection does work. However, the generalization of current ER methods to evolve controllers capable of performing more difficult tasks may require the development of new approaches to fitness evaluation.

4.3. Co-evolution of controller and morphology

In several evolutionary robotics papers, it has been suggested that the simultaneous evolution of morphology and control provides a pathway toward the development of complex robots expressing complex behaviors, and this holds some promise. However, the underlying argument that the co-evolution of body and mind is the only way to generate a complex controller fitted to a complex body is not entirely supported by the literature or by observation. Humans are much better at designing physical systems than they are at designing intelligent control systems: complex powered machinery has been in existence for over 150 years, whereas it is safe to say that no truly intelligent autonomous machine has ever been built by a human. There is also genetic evidence from genome analysis that indicates natural evolution of cognitive function is evolving more rapidly than anatomical structure in modern humans [86]. This implies that there may be a lag in the evolution of intelligence in humans compared to physical development.

The co-evolution of bodies and minds does have potential though, and several recent works have overcome previous barriers by combining high-fidelity simulation environments with modular elemental component libraries [16,65,68]. These newer co-evolution works have used aggregate or very low bias selection methods, and complex physical robots were fabricated and tested in the real world.

4.4. The role of survival and fitness assessment in simulation

If fitness could be implemented simply as the ability to propagate within a complex environment, it is possible that systems with novel integrated behavioral intelligences could be evolved. In order to achieve this, the fundamental element of survival must be formulated in a way that is consistent with the actual physical representation of the environment. Specifically, robots should fail to survive when they physically stop functioning. It is not currently feasible to perform embodied evolutions of this type because many robots would be damaged and destroyed during evolution. For example, a population of 50 robots evolved for 100 generations with 50% failure during each generation would generate 2500 fatal robot failures. At the present time, the only alternative is simulation.

The field of artificial life (AL) studies this concept of evolution based on survival in simulated environments, but AL research very often uses artificial measures of survival (essentially objective functions), such as the ability to gather food or energy icons. These measures of survival are not consistent with the underlying representations of the simulated agents, or the simulated physics of the environments.

Simulations used in evolutionary robotics attempt to consistently integrate all necessary physical elements of the robot's environment. For example, in the peg collecting behavior evolved in [14], the pegs were simulated as physical objects, fully integrated into the object representation of the simulated environment. The fidelity of the simulation was verified by testing evolved controllers in the real world. Here, though, fitness was defined in terms of success at collecting pegs, rather than physical survival of the robots. In the competitive goal homing task studied in [21], robots, obstacles and target objects were integrated into the simulation environment using representations that were consistent with a physical environment. However, survival, enforced by the fitness function, was couched in terms of robots physically arriving at the goal locations in the environment, and not in terms of actual physical survival of the robot. Although the experiments discussed in [14,21] produced robots capable of performing their intended tasks, neither of these research efforts produced robots capable of physical survival in a complex environment *per se*.

In order for autonomous robot simulation environments to be useful in studying real evolutionary processes, and in evolving true survival abilities in robots above and beyond performing simple well-defined tasks, they must include measures of survival that are truly integrated into the fabric of the simulation environment, while at the same time being consistent with real robotic systems. Evolutionary robotics has not reached this stage of generality, and this will be an important area of research. Survival in complex environments is a fundamental goal in evolutionary robotics and autonomous intelligent robotics as a whole.

In many ways the concept of fitness to perform a specific task is at odds with fundamental survival in a complex environment. If adaptation to environmental conditions is to be studied in the general case, specific tasks should not be imposed upon the evolving systems. Developmental robotics [108,109] addresses the problem of environmental-based learning by attempting to endow robots with environmentally stimulated self-organizing abilities or environmentally stimulated novelty-seeking. This eliminates the need for an explicit objective function to drive the learning process, but does not necessarily result in the learning of any one particular behavior, beyond negotiating a particular environment.

4.5. The long-term prospects for evolving intelligence

Is it possible to automatically generate controllers for arbitrarily difficult and complex tasks in the general case? The short answer to this question is probably no. The tasks currently investigated by evolutionary robotics are relatively simple compared to natural systems and shed little light on the question of learning arbitrarily difficult tasks. Even natural intelligent systems are limited in their degree of sophistication, at least as represented by life forms observed on Earth. To make matters worse, the overall abilities of complex animals and humans are difficult to describe, much less duplicate in a holistic way.

Let us put some bounds on this general question of using artificial evolution to generating controllers for arbitrarily difficult tasks: is it possible to use artificial evolution to generate intelligent systems capable of solving large classes of tasks in the realm of intelligent autonomous systems? The answer to this question is unknown, but current evolutionary robotics results indicate that it may be possible to generate autonomous systems with limited general abilities at some point in the future.

Current ER research has demonstrated that competent autonomous environmentally situated agents can be evolved. The most complex evolved robot systems are capable of achieving three or four interconnected abilities in a coordinated fashion that allows an overall task to be completed. These results include the multiple sequential goal homing task [22], the object collection and deposition task [14], and the competitive team searching task [21], all discussed in the survey section of this paper. These systems were generated based on feedback from environmental interaction and represent a step toward more general systems.

5. Conclusion

In this paper we have reviewed the use of fitness functions in the field of evolutionary robotics. Many representative works were selected from the literature and the evolutionary processes used were summarized in terms of the fitness functions. Functions were reported using a standardized nomenclature to aid in comparison. It was found that much of the research made use of fitness functions that were selective for solutions that the researchers had envisioned before the initiation of the evolutionary processes. The degree to which features of evolved solutions reflected *a priori* knowledge on the parts of the human researchers varied. A portion of the reviewed research reported the evolution of controllers which demonstrated novel abilities not specifically defined by the fitness functions used during evolution. Recent work done in the last few years has begun to use aggregate fitness selection methods that introduce much less human knowledge and bias into the evolved controllers.

The fundamental question of how to select for truly complex intelligent autonomous behaviors during evolution remains largely unanswered and evolutionary robotics remains somewhat on the fringes of autonomous robotics research. It should be pointed out that other non-evolutionary computing-based attempts at producing robots capable of learning novel intelligent control have, in general, stumbled up against the same problem of how to drive the learning process without introducing an essentially *a priori* known control strategy into the learned systems.

Acknowledgements

The authors would like to thank Brenae Bailey for editorial input, insightful conversations, and support in preparing this manuscript.

References

[1] M. Mataric, D. Cliff, Challenges in evolving controllers for physical robots, Robotics and Autonomous Systems 19 (1) (1996) 67–83.
[2] I. Harvey, P. Husbands, D. Cliff, A. Thompson, N. Jakobi, Evolutionary robotics: The Sussex approach, Robotics and Autonomous Systems 20 (2–4) (1997) 205–224.
[3] L.A. Meeden, D. Kumar, Trends in evolutionary robotics, in: L.C. Jain, T. Fukuda (Eds.), Soft Computing for Intelligent Robotic Systems, Physica-Verlag, New York, NY, 1998, pp. 215–233.
[4] S. Nolfi, D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines, The MIT Press, Cambridge, Massachusetts, 2000.
[5] J. Walker, Simon Garrett, M. Wilson, Evolving controllers for real robots: A survey of the literature, Adaptive Behavior 11 (3) (2003) 179–203.
[6] D.K. Pratihar, Evolutionary robotics—A review, Sadhana - Academy Proceedings in Engineering Sciences 28 (6) (2003) 999–1011.
[7] K.C. Tan, L.F. Wang, T.H. Lee, P. Vadakkepat, Evolvable hardware in evolutionary robotics, Autonomous Robots 16 (1) (2004) 5–21.
[8] J. Teo, H.A. Abbass, Multiobjectivity and complexity in embodied cognition, IEEE Transactions on Evolutionary Computation 9 (4) (2005) 337–360.
[9] I. Harvey, P. Husbands, D. Cliff, Seeing the light: Artificial evolution, real vision, in: D. Cliff, P. Husbands, J.-A. Meyer, S. Wilson (Eds.), From Animals to Animates 3, Proc. of 3rd Intl. Conf. on Simulation of Adaptive Behavior, SAB94, MIT Press/Bradford Books, Boston, MA, 1994, pp. 392–401.
[10] S. Nolfi, D. Floreano, Co-evolving predator and prey robots: Do 'arms races' arise in artificial evolution? Artificial Life 4 (4) (1998) 311–335.
[11] D. Floreano, F. Mondada, Evolution of homing navigation in a real mobile robot, IEEE Transactions on Systems, Man, Cybernetics Part B: Cybernetics 26 (3) (1996) 396–407.
[12] R.A. Watson, S.G. Ficici, J.B. Pollack, Embodied evolution: Distributing an evolutionary algorithm in a population of robots, Robotics and Autonomous Systems 39 (1) (2002) 1–18.

[13] H.H. Lund, O. Miglino, From simulated to real robots, in: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 362–365.
[14] S. Nolfi, Evolving non-trivial behaviors on real robots, Robotics and Autonomous Systems 22 (3–4) (1997) 187–198.
[15] P. Nordin, W. Banzhaf, M. Brameier, Evolution of a world model for a miniature robot using genetic programming, Robotics and Autonomous Systems 25 (1–2) (1998) 105–116.
[16] H. Lipson, J.B. Pollack, Automatic design and manufacture of robotic lifeforms, Nature 406 (6799) (2000) 974–978.
[17] G.S. Hornby, S. Takamura, J. Yokono, O. Hanagata, M. Fujita, J. Pollack, Evolution of controllers from a high-level simulator to a high dof robot, in: J. Miller (Ed.), Evolvable Systems: From Biology to Hardware; Proceedings of the Third International Conference, ICES 2000, in: Lecture Notes in Computer Science, vol. 1801, Springer, 2000, pp. 80–89.
[18] A.L. Nelson, E. Grant, G.J. Barlow, T.C. Henderson, A colony of robots using vision sensing and evolved neural controllers, in: Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS03, Las Vegas NV, 27–31 Oct., 2003, pp. 2273–2278.
[19] F. Gomez, R. Miikkulainen, Transfer of neuroevolved controllers in unstable domains, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-04, Seattle, WA, 2004, pp. 957–968.
[20] D. Floreano, J. Urzelai, Evolutionary robots with on-line self-organization and behavioral fitness, Neural Networks 13 (4–5) (2000) 431–443.
[21] A.L. Nelson, E. Grant, Using direct competition to select for competent controllers in evolutionary robotics, Robotics and Autonomous Systems 54 (10) (2006) 840–857.
[22] G. Capi, K. Doya, Evolution of recurrent neural controllers using an extended parallel genetic algorithm, Robotics and Autonomous Systems 52 (2–3) (2005) 148–159.
[23] I. Cloete, J.M. Zurada (Eds.), Knowledge-Based Neurocomputing, The MIT Press, Cambridge, MA, London, England, ISBN: 0-262-03274-0, 2000.
[24] C. Dima, M. Herbert, A. Stentz, Enabling learning from large datasets: Applying active learning to mobile robotics, in: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA, New Orleans, LA, 2004, pp. 108–114.
[25] A.L. Nelson, E. Grant, G. Lee, Using genetic algorithms to capture behavioral traits exhibited by knowledge based robot agents, in: Proceedings of the ISCA 15th International Conference: Computer Applications in Industry and Engineering, CAINE-2002, San Diego, CA, 7–9 Nov., 2002, pp. 92–97.
[26] W. Banzhaf, P. Nordin, M. Olmer, Generating adaptive behavior using function regression within genetic programming and a real robot, in: Proceedings of the Second International Conference on Genetic Programming, San Francisco, 1997, pp. 35–43.
[27] N. Jakobi, Running across the reality gap: Octopod locomotion evolved in a minimal simulation, in: P. Husbands, J.A. Meyer (Eds.), Evolutionary Robotics: First European Workshop, EvoRobot98, Springer-Verlag, 1998, pp. 39–58.
[28] A.C. Schultz, J.J. Grefenstette, W. Adams, RoboShepherd: Learning a complex behavior, Robotics and Manufacturing: Recent Trends in Research and Applications 6 (1996) 763–768.
[29] D. Keymeulen, M. Iwata, Y. Kuniyoshi, T. Higuchi, Online evolution for a self-adapting robotic navigation system using evolvable hardware, Artificial Life 4 (4) (1998) 359–393.
[30] M. Quinn, L. Smith, G. Mayley, P. Husbands, Evolving team behaviour for real robots, in: EPSRC/BBSRC International Workshop on Biologically-Inspired Robotics: The Legacy of W. Grey Walter, WGW'02, 14–16 Aug., 2002, HP Bristol Labs, UK.
[31] K. Kawai, A. Ishiguro, P. Eggenberger, Incremental evolution of neuro-controllers with a diffusion-reaction mechanism of neuromodulators, in: Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'01, vol. 4, Maui, HI, Oct. 29–Nov. 3, 2001, pp. 2384–2391.
[32] F. Hoffmann, J.C.S. Zagal Montealegre, Evolution of a tactile wall-following behavior in real time, in: The 6th Online World Conference on Soft Computing in Industrial Applications, WSC6, 10–24 Sept., 2001.
[33] W. Lee, J. Hallam, H. Lund, Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots, in: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, 1997, pp. 495–499.
[34] G.J. Barlow, L.S. Mattos, E. Grant, C.K. Oh, Transference of evolved unmanned aerial vehicle controllers to a wheeled mobile robot, in: Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 2005.
[35] H. Nakamura, A. Ishiguro, Y. Uchilkawa, Evolutionary construction of behavior arbitration mechanisms based on dynamically-rearranging neural networks, in: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 1, IEEE, 2000, pp. 158–165.
[36] F. Pasemann, U. Steinmetz, M. Hülse, B. Lara, Evolving brain structures for robot control, in: IWANN'01 Proceedings LNCS, 2085, vol. II, Springer-Verlag, Granada, Spain, 2001, pp. 410–417.
[37] O Miglino, D. Denaro, G. Tascini, D Parisi, Detour behavior in evolving robots: Are internal representations necessary? in: Proceedings of the First European Workshop on Evolutionary Robotics, Springer-Verlag, 1998, pp. 59–70.
[38] G. Buason, N. Bergfeldt, T. Ziemke, Brains, bodies, and beyond: Competitive co-evolution of robot controllers, morphologies and environments, Genetic Programming and Evolvable Machines 6 (1) (2005) 25–51.
[39] D. Cliff, G.F. Miller, Co-evolution of pursuit and evasion II: Simulation methods and results, in: P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, S.W. Wilson (Eds.), From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, SAB96, MIT Press, Bradford Books, 1996, pp. 506–515.
[40] K. Chellapilla, D.B. Fogel, Evolving an expert checkers playing program without using human expertise, IEEE Transactions on Evolutionary Computation 5 (4) (2001) 422–428.
[41] A. Lubberts, R. Miikkulainen, Co-evolving a go-playing neural network, in: Coevolution: Turning Algorithms upon Themselves, Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conference, GECCO-2001, San Francisco, CA, 2001.
[42] T. Gomi, K. Ide, Evolution of gaits of a legged robot, in: The 1998 IEEE International Conference on Fuzzy Systems, Proceedings of the 1998 IEEE World Congress on Computational Intelligence, vol. 1, 4–9 May, 1998, pp. 159–164.
[43] V. Matellán, C. Fernández, J.M. Molina, Genetic learning of fuzzy reactive controllers, Robotics and Autonomous Systems 25 (1–2) (1998) 33–41.
[44] J. Liu, C.K. Pok, H.K. Keung, Learning coordinated maneuvers in complex environments: A sumo experiment, in: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, vol. 1, 6–9 July, 1999, pp. 343–349.
[45] H.-S. Seok, K.-J. Lee, J.-G. Joung, B.-T. Zhang, An on-line learning method for object-locating robots using genetic programming on evolvable hardware, in: Proceedings of the Fifth International Symposium on Artificial Life and Robotics, AROB'00, vol. 1, 2000, pp. 321–324.
[46] J. Ziegler, W. Banzhaf, Evolving control metabolisms for a robot, Artificial Life 7 (2) (2001) 171–190.
[47] F. Hoffmann, G. Pfister, Evolutionary learning of a fuzzy control rule base for an autonomous vehicle, in: Proceedings of the Fifth International Conference IPMU: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain, July 1996, pp. 659–664.
[48] A. Thompson, Evolving electronic robot controllers that exploit hardware resources, Advances in Artificial Life: Proceedings of the 3rd European Conference on Artificial Life, ECAL95, Lausanne, vol. 929, Springer-Verlag, 1995, pp. 640–656.
[49] A. Ishiguro, S. Tokura, T. Kondo, Y. Uchikawa, Reduction of the gap between simulated and real environments in evolutionary robotics: A dynamically-rearranging neural network approach, in: Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics, vol. 3, 1999, pp. 239–244.
[50] M. Ebner, A. Zell, Evolving a behavior-based control architecture — From simulations to the real world, in: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, 13–17 July 1999, pp. 1009–1014.
[51] W. Lee, Evolving autonomous robot/sensors: From controller to morphology, IEICE Transactions of Information and Systems E83-D (2) (2000) 200–210.
[52] I.G. Sprinkhuizen-Kuyper, R. Kortmann, E.O. Postma, Fitness functions for evolving box-pushing behaviour, in: A. van den Bosch, H. Weigand (Eds.), Proceedings of the Twelfth Belgium–Netherlands Artificial Intelligence Conference, 2000, pp. 275–282.
[53] K. Wolff, P. Nordin, Evolution of efficient gait with an autonomous biped robot using visual feedback, in: Proceedings of the 2nd IEEE-RAS International Conference on Humanoid Robots, Humanoids 2001, Tokyo, Japan, 2001, pp. 99–106.
[54] U. Nehmzow, Physically embedded genetic algorithm learning in multi-robot scenarios: The PEGA algorithm, in: Proceedings of the Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Edinburgh, 2002.
[55] E.D.V. Simões, D.A.C. Barone, Predation: An approach to improving the evolution of real robots with a distributed evolutionary controller, in: Proceedings of the IEEE International Conference on Robot Automation, ICRA'02, Washington, DC, May, 2002, pp. 664–669.
[56] D. Marocco, D. Floreano, Active vision and feature selection in evolutionary behavioral systems, in: J. Hallam, D. Floreano, G. Hayes, J. Meyer (Eds.), From Animals to Animats 7, MIT Press, Cambridge, MA, 2002.
[57] M. Okura, A. Matsumoto, H. Ikeda, K. Murase, Artificial evolution of FPGA that controls a miniature mobile robot Khepera, in: Proceedings of the Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE 2003), 4–6 Aug., 2003, vol. 3 pp. 2858–2863.
[58] D. Gu, H. Hu, J. Reynolds, E. Tsang, GA-based learning in behaviour based robotics, in: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Kobe, Japan, 16–20 July, 2003, vol. 3, pp. 1521–1526.
[59] A.L. Nelson, E. Grant, J.M. Galeotti, S. Rhody, Maze exploration behaviors using an integrated evolutionary robotics environment, Journal of Robotics and Autonomous Systems 46 (3) (2004) 159–173.
[60] A. Boeing, S. Hanham, T. Braunl, Evolving autonomous biped control from simulation to reality, in: Proceedings of the 2nd International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand, 13–15 Dec., 2004, pp. 440–445.
[61] G.S. Hornby, S. Takamura, T. Yamamoto, M. Fujita, Autonomous evolution of dynamic gaits with two quadruped robots, IEEE Transactions on Robotics 21 (3) (2005) 402–410.
[62] S. Kamio, H. Iba, Adaptation technique for integrating genetic programming and reinforcement learning for real robots, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 318–333.

[63] G.B. Parker, R. Georgescu, Using cyclic genetic algorithms to evolve multi-loop control programs, in: Proceedings of the 2005 IEEE International Conference on Mechatronics and Automation, ICMA 2005, July 2005, Niagara Falls, Ontario, Canada.
[64] E.J.P. Earon, T.D. Barfoot, G.M.T. D'Eleuterio, From the sea to the sidewalk: the evolution of hexapod walking gaits by a genetic algorithm, in: Proceedings of the International Conference on Evolvable Systems, ICES, Edinburgh, Scotland, 17–19 April, 2000.
[65] G.S Hornby, H. Lipson, J.B. Pollack, Evolution of generative design systems for modular physical robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'01, vol. 4, 2001, pp. 4146–4151.
[66] P. Augustsson, K. Wolff, P. Nordin, Creation of a learning, flying robot by means of evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002, New York, 9–13 July, 2002, Morgan Kaufmann, 2002, pp. 1279–1285.
[67] J. Zufferey, D. Floreano, M. van Leeuwen, T. Merenda, Evolving vision based flying robot, in: Bülthoff, Lee, Poggio, Wallraven (Eds.), Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision, in: LNCS, vol. 2525, Springer-Verlag, Berlin, pp. 592–600.
[68] I. Macinnes, E. Di Paolo, Crawling out of the simulation: Evolving real robot morphologies using cheap, reusable modules, in: Proceedings of the International Conference on Artificial Life, ALIFE9, Boston, MA, 12–15 Sept., MIT Press, 2004, pp. 94–99.
[69] V. Zykov, J. Bongard, H. Lipson, Evolving dynamic gaits on a physical robot, in: 2004 Genetic and Evolutionary Computation Conference, GECCO, Seattle, WA, 2004.
[70] S. Chernova, M. Veloso, An evolutionary approach to gait learning for four-legged robots, in: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, IROS'04, vol. 3, Sendai, Japan, Sept. 28–Oct. 2, 2004, pp. 2562–2567.
[71] D. Filliat, J. Kodjabachian, J.A. Meyer, Incremental evolution of neural controllers for navigation in a 6 1egged robot, in: Sugisaka, Tanaka (Eds.), Proceedings of the Fourth International Symposium on Artificial Life and Robotics, Oita Univ. Press, 1999.
[72] F. Gruau, Automatic definition of modular neural networks, Adaptive Behavior 2 (1995) 151–183.
[73] M. Quinn, Evolving communication without dedicated communication channels, in: J. Kelemen, P. Sosik (Eds.), Advances in Artificial Life: Sixth European Conference on Artificial Life, ECAL 2001, Prague, Czech Republic, Sept. 2001, Springer, 2001, pp. 357–366.
[74] T. Ziemke, Remembering how to behave: Recurrent neural networks for adaptive robot behavior, in: Medsker, Jain (Eds.), Recurrent Neural Networks: Design and Applications, CRC Press, Boca Raton, 1999.
[75] E. Tuci, M. Quinn, I. Harvey, Evolving fixed-weight networks for learning robots, in: Proc. 2002 Congress on Evolutionary Computing, Honolulu HI, vol. 2, 2002, pp. 1970–1975.
[76] I. Ashiru, C.A. Czarnecki, Evolving communicating controllers for multiple mobile robot systems, in: Proceedings of the 1998 IEEE International Conference on Robotics and Automation, vol. 4, 1998, pp. 3498–3503.
[77] M. Quinn, Evolving cooperative homogeneous multi-robot teams, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'00, vol. 3, Takamatsu Japan, 2000, pp. 1798–1803.
[78] G. Baldassarre, S. Nolfi, D. Parisi, Evolving mobile robots able to display collective behaviors, in: C.K. Hemelrijk, E. Bonabeau (Eds.), Proceedings of the International Workshop on Self-Organisation and Evolution of Social Behaviour, Monte Verità, Ascona, Switzerland, 8–13 Sept., 2002, pp. 11–22.
[79] M. Dorigo, V. Trianni, E. Sahin, T. Labella, R. Grossy, G. Baldassarre, S. Nolfi, J.-l. Deneubourg, F. Mondada, D. Floreano, L. Gambardella, Evolving self-organizing behaviors for a swarm-bot, Autonomous Robots 17 (23) (2004) 223–245.
[80] R. Groß, M. Bonani, F. Mondada, M. Dorigo, Autonomous self-assembly in a swarmbot, in: K. Murase, K. Sekiyama, N. Kubota, T. Naniwa, J. Sitte (Eds.), Proceedings of the Third International Symposium on Autonomous Minirobots for Research and Edutainment, Springer Verlag, Berlin, 2006, pp. 314–322.
[81] J.A. Driscoll, R.A. Peters II, A development environment for evolutionary robotics, in: 2000 IEEE International Conference on Systems, Man, and Cybernetics, vol, 5, 2000, pp. 3841–3845.
[82] F. Gomez, R. Miikkulainen, Incremental evolution of complex general behavior, Adaptive Behavior 5 (1997) 317–342.
[83] G.J. Barlow, C.K. Oh, E. Grant, Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming, in: Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems, CIS, Singapore, Dec. 2004, pp. 688–693.
[84] D. Cliff, G.F. Miller, Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations, in: F. Moran, A. Moreno, J.J. Merelo, P. Cachon (Eds.), Proceedings of the Third European Conference on Artificial Life: Advances in Artificial Life, ECAL95, in: Lecture Notes in Artificial Intelligence, vol. 929, Springer-Verlag, 1995, pp. 200–218.
[85] W. Lee, J. Hallam, H.H. Lund, A hybrid GP/GA approach for co-evolving controllers and robot bodies to achieve fitness-specified task, in: Proceedings of the IEEE 3rd International Conference on Evolutionary Computation, 20–22 May, 1996, pp. 384–389.
[86] N. Mekel-Bobrov, S.L. Gilbert, P.D. Evans, E.J. Vallender, J.R. Anderson, R.R. Hudson, S.A. Tishkoff, B.T. Lahn, Ongoing adaptive evolution of ASPM, a brain size determinant in homo sapiens, Science 309 (5741) (2005) 1720–1722.
[87] J. Walker, S. Garrett, M. Wilson, The balance between initial training and lifelong adaptation in evolving robot controllers, IEEE Transactions on Systems, Man and Cybernetics 36 (2) (2006) 423–432.
[88] L.E. Parker, ALLIANCE: An architecture for fault tolerant multi-robot cooperation, IEEE Transactions on Robotics and Automation 14 (2) (1998) 220–240.
[89] A. Orebäck, H.I. Christensen, Evaluation of architectures for mobile robotics, Autonomous Robots 14 (1) (2003) 33–49.
[90] T.M. Mitchell, Machine Learning, McGraw Hill, 1997.
[91] W.L. Brogan, Modern Control Theory, third ed., Prentice-Hall, NJ, 1991.
[92] S. Thrun, Robotic mapping: A survey, in: G. Lakemeyer, B. Nebel (Eds.), Exploring Artificial Intelligence in the New Millennium, Morgan Kaufmann, 2002.
[93] S. Thrun, Bayesian landmark learning for mobile robot localization, Machine Learning 33 (1) (1998) 41–76.
[94] K. Dixon, P. Khosla, Learning by observation with mobile robots: a computational approach, in: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA'04, New Orleans, LA, 2004, pp. 102–107.
[95] M. Mitchell, An Introduction to Genetic Algorithms, The MIT Press, Cambridge, Massachusetts, 1998.
[96] J. Galeotti, S. Rhody, A.L. Nelson, E. Grant, G. Lee, EvBots – The design and construction of a mobile robot colony for conducting evolutionary robotic experiments, in: Proceedings of the ISCA 15th International Conference: Computer Applications in Industry and Engineering, CAINE-2002, San Diego, CA, 7–9 Nov., 2002, pp. 86–91.
[97] F. Mondada, E. Franzi, P. Ienne, Mobile robot miniaturization: A tool for investigation in control algorithms, in: The 3rd International Symposium on Experimental Robotics, ISER'93, Kyoto, Japan, October 1993, in: Lecture Notes in Control and Information Sciences, vol. 200, 1993, pp. 501–513.
[98] T.M.C. Smith, P. Husbands, P. Layzell, M. O'Shea, Fitness landscapes and evolvability, Evolutionary Computation 10 (1) (2002) 1–34.
[99] F. Kaplan, P. Oudeyer, E. Kubinyi, A. Miklosi, Robotic clicker training, Robotics and Autonomous Systems 38 (3–4) (2002) 197–206.
[100] H.H. Lund, O. Miglino, L. Pagliarini, A. Billard, A. Ijspeert, Evolutionary robotics—A children's game, in: Evolutionary Computation Proceedings, 1998 IEEE World Congress on Computational Intelligence, 1998, pp. 154–158.
[101] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man and Cybernetics, Part B 26 (1) (1996) 29–41.
[102] L.N. de Castro, J. Timmis, Artificial immune systems as a novel soft computing paradigm, Soft Computing 7 (8) (2003) 526–544.
[103] J.A. Meyer, A. Guillot, Simulation of adaptive behavior in animats: Review and prospect, in: J.A. Meyer, S. Wilson (Eds.), From Animals to Animats, Proceedings of the First International Conference on Simulation of Adaptive Behavior, SAB-90, MIT Press, 1991, pp. 2–14.
[104] A. Guillot, J.A. Meyer, The animat contribution to cognitive systems research, Journal of Cognitive Systems Research 2 (2) (2001) 157–165.
[105] G.J. Barlow, C.K. Oh, Robustness analysis of genetic programming controllers for unmanned aerial vehicles, in: Proceedings of the 8th Annual Conference on Genetic And Evolutionary Computation, GECCO'06, 2006, pp. 135–142.
[106] M. Kaiser, H. Friedrich, R. Buckingham, K. Khodabandehloo, S. Tomlinson, Towards a general measure of skill for learning robots, in: Proceedings of the 5th European Workshop on Learning Robots, Bari, Italy, 1996.
[107] A. Nelson, E. Grant, Aggregate selection in evolutionary robotics, in: N. Nedjah, L. Coelho, L. Mourelle (Eds.), Mobile Robots: The Evolutionary Approach, in: Studies in Computational Intelligence, vol. 50, Springer, 2007, pp. 63–88.
[108] M. Lungarella, G. Metta, R. Pfeifer, G. Sandini, Developmental robotics: A survey, Connection Science 15 (4) (2003) 151–190.
[109] M. Asada, K.F. MacDorman, H. Ishiguro, Y. Kuniyoshi, Cognitive developmental robotics as a new paradigm for the design of humanoid robots, Robotics and Autonomous Systems 37 (2–3) (2001) 185–193.
[110] V. Trianni, M. Dorigo, Self-organisation and communication in groups of simulated and physical robots, Biological Cybernetics 95 (3) (2006) 213–231.

Dr. Andrew L. Nelson was born in Laramie, Wyoming in 1967. He received his B.S. degree with specialization in Computer Science from the Evergreen State College in Olympia Washington in 1990. He received his M.S. in Electrical Engineering from North Carolina State University in 2000. He received his Ph.D. in Electrical Engineering at the Center for Robotics and Intelligent Machines (CRIM) at North Carolina State University in 2003. Between 2003 and 2005 he was a visiting researcher at the University of South Florida. Currently he is a researcher at Androtics LLC, Tucson AZ and Santa Cruz CA. His main interests are in the fields of fully autonomous robot control, bio-inspired robot control and evolutionary robotics. His robotics work has included applying artificial evolution to synthesize controllers for swarms of autonomous robots as well as the development of fuzzy-logic-based controllers for robot navigation. He pursues work in artificial neural networks, genetic algorithms and soft computing related to autonomous machine control. He has also conducted research in diverse fields including electric machine design and molecular biology.

Gregory J. Barlow is a Ph.D. candidate at the Robotics Institute at Carnegie Mellon University. He received B.S. degrees in electrical and computer engineering in 2003 from North Carolina State University in 2003 and an M.S. degree in electrical engineering from North Carolina State University in 2004. His research interests include memory-enhanced algorithms for dynamic problems, dynamic optimization with evolutionary algorithms, and evolutionary robotics.

Dr. Lefteris Doitsidis received his B.S. degree from the Production Engineering and Management Department of the Technical University of Crete, Chania, Greece, in 2000. In 2002 he was awarded his M.S degree in Production Systems and in 2008 he received his Ph.D. at the same department. Since 2002, he has been a researcher at the Intelligent Systems and Robotics Laboratory of the same department. From August 2003 to June 2004 he was a visiting scholar at the University of South Florida, FL, U.S.A. Beginning in 2004 he has been, and continues to be an instructor at the Technological Educational Institute of Crete. His research interests lie in the areas of multirobot teams, autonomous operation and navigation of unmanned vehicles and evolutionary computation.