A. Nelson, E. Grant, "Aggregate selection in evolutionary robotics," in *Mobile Robots: The Evolutionary Approach*, eds. N. Nedjah, L. Coelho, L. Mourelle, Studies in Computational Intelligence , Vol. 50, pp. 63-88, Springer, 2007.

AGGREGATE SELECTION IN EVOLUTIONARY ROBOTICS

Andrew L. Nelson
Androtics LLC, P.O. Box 44065, Tucson, AZ 85733-4065 USA
(phone: 520-822-6921; e-mail: alnelson@ieee.org, web: www.nelsonrobotics.org)

Edward Grant
Center for Robotics and Intelligent Machines
North Carolina State University, Raleigh, NC 27695 USA
(phone: 919-513-0787; e-mail: egrant@ncsu.edu, web: http://www.crim.ncsu.edu/)

Can the processes of natural evolution be mimicked to create robots or autonomous agents? This question embodies the most fundamental goals of evolutionary robotics (ER). ER is a field of research that explores the use of artificial evolution and evolutionary computing for learning of control in autonomous robots, and in autonomous agents in general.

In a typical ER experiment, robots, or more precisely their control systems, are evolved to perform a given task in which they must interact dynamically with their environment. Controllers compete in the environment and are selected and propagated based on their ability (or fitness) to perform the desired task. A key component of this process is the manner in which the fitness of the evolving controllers is measured.

In ER, fitness is measured by a fitness function or objective function. This function applies some given criteria to determine which robots or agents are better at performing the task for which they are being evolved. Fitness functions can introduce varying levels of *a priori* knowledge into evolving populations. Some types of fitness functions encode the important features of a known solution to a given task. Populations of controllers evolved using such functions then reproduce these features and essentially evolve control systems that duplicate an *a priori* known algorithm. In contrast to this, evolution can also be performed using a fitness function that incorporates no knowledge of how the particular task at hand is to be achieved. In these cases all selection is based only on whether robots/agents succeed or fail to complete the task. Such fitness functions are referred to as *aggregate* because they combine the benefit or deficit of all actions a given agent performs into a single success/failure term.

Fitness functions that select for specific solutions do not allow for fundamentally novel control learning. At best, these fitness functions perform some degree of optimization, and provide a method for transferring known control heuristics to robots. At some level, selection must be based on a degree of overall task completion independent of particular behaviors or task solution features if true learning rather than simple optimization or transference is to be achieved.

Aggregate fitness functions measure overall task completion. However, they can suffer from an inability to produce non-random selection in nascent unevolved populations. If the task is too difficult, it is likely that none of the

randomly initialized controllers will be able to make any meaningful progress toward completing the overall task.

This chapter investigates how aggregate fitness functions have been and continue to be used in ER, what levels of success they have generated relative to other fitness measurement methods, and how problems with them might be overcome.

I. INTRODUCTION

A distinction can be made between what a robot does, and how it does it. That is, there is a difference between the task that a robot is to perform, and the manner in which it performs or solves the task. For example, consider a robot that is to be designed to move toward a light source (phototaxis). The robot's task is phototaxis, but there are many ways in which this task could be performed. For instance, the robot might detect the light source, turn toward it, and then move forward until it collides with the source. Another solution might be for the robot to just wander around in its environment until it detected a threshold magnitude of light indicating it was near the source, at which point it would stop.

In general there are many solutions (of varying quality) to any given task. Determining the relative qualities of different solutions is essential for control learning. In artificial evolution-based forms of learning, fitness functions make this determination in an automatic or algorithmic way. The distinction between task and task solution defines two broad classes of fitness functions, namely *behavioral* fitness functions and *aggregate* fitness functions, and these will be a central focus of discussion in the following sections of this chapter.

Most autonomous robot systems are currently programmed by hand to perform their intended tasks. Learning to perform non-trivial tasks remains a largely unsolved problem in autonomous robotics. Evolutionary robotics approaches the problem of autonomous control learning through population-based artificial evolution. ER methods bear a great deal of similarity to other approaches to controller learning in autonomous robots. In particular, most learning methods require an objective function, and although the discussion in this chapter is focused on experimental work that involved population-based learning, most of the issues related to fitness evaluation are directly relevant to any autonomous system that is intended to learn basic control or behavior in a dynamic environment. We should point out here that other applications of machine learning in robots that are not aimed at learning the primary dynamic control of an agent, such as object recognition, mapping, or trajectory tracking, are generally amenable to heuristic algorithmic methods, or example-based gradient descent learning methods. These methods are not directly applicable to systems intended to learn how to perform complex tasks autonomously in dynamic environments.

To date, ER research has almost exclusively focused its attention on a handful of benchmark robot behavioral tasks, including phototaxis, locomotion and object avoidance, foraging, and goal homing. A large portion of this research has used fitness functions that selected for particular task solution elements that were

known *a priori* to allow the task to be accomplished. Because of the simplicity of the tasks investigated, many of the fundamental problems associated with using fitness functions that contain *a priori* task solution information have not become apparent. If there are only a few near-optimal solutions to a particular problem, it is less obvious that a particular fitness function might have forced the evolution of some of the features of a resulting solution. The focus on a few simple tasks has also downplayed problems associated with the application of aggregate selection, to a degree. If a task is simple enough, an aggregate fitness function is likely to be able to drive evolution effectively starting from a randomly initialized population with no special treatment.

An important unanswered question within the field of ER is whether the methods used up to this point can be generalized to produce more sophisticated truly non-trivial autonomous robot control systems. Successful evolution of intelligent autonomous robot controllers is ultimately dependent on obtaining suitable fitness functions that are capable of selecting for task competence without specifying the low-level implementation details of those behaviors.

The remainder of this chapter is organized as follows: the rest of this section provides a general summary of ER methodology and related terms, and lays the foundation for discussion in later sections.

In Sections II and III we very briefly review the field of ER research in general and then provide a more in-depth review of ER work in which aggregate or nearly aggregate selection mechanisms were used.

In Section IV we discuss methods for overcoming difficulties associated with using aggregate fitness functions to evolve populations of mobile robot controllers for specific tasks.

In section V we provide an overview of our own work, which investigated methods for overcoming difficulties associated with aggregate selection in evolutionary robotics.

In section VI we conclude the article with a discussion of the long-term prospects of using artificial evolution to generate complex autonomous mobile robot controllers.

A. Evolutionary Robotics Process Overview

In a typical ER experiment, artificial evolution and genetic algorithms are used to create robot or agent controllers able to perform some sort of autonomous task or behavior. The artificial evolution process starts with a randomly initialized population of robot or agent controllers. Over a series of generations or trial evaluation periods robots compete against one another to perform a given task. Robots that perform the task better are selected and their control systems are duplicated and altered using operations inspired by natural mutation and recombination. The altered controllers (the *offspring*) then replace the controllers of the poorly performing robots in the population. This process is iterated many times until a suitably fit controller arises.

It is possible to seed initial populations with controllers that are not randomly configured. Such populations might come from a previous evolutionary process

(incremental evolution [29]), from a hand-directed training method (clicker or breeder training [53]), or could have been initially configured by some other means. However, such previously configured seed populations contain biases, and these biases can have an effect on the course of later evolution.

B. Bias

Before entering into a description of types of fitness functions used in ER, it makes sense to say a word or two about bias and sources of bias in evolving systems.

In the most general sense, bias refers to a tendency of a given dynamic system to develop toward a particular state. Bias in this context can be thought of as a form of pressure acting to change or evolve a system toward a particular quality or form. We use the terms *primary* and *secondary bias*.

Primary bias (or *representation bias*) describes any bias introduced due to a system's fundamental representation. Tendencies or abilities of a system due to the fundamental rules that describe the system (as in a simulation) are forms of representation bias.

Secondary biases are those that are imposed on a given system from outside the system in a way that is not necessarily consistent with the underlying representation of the system. Selection mechanisms in the form of fitness functions in artificial evolutionary systems represent secondary biases. As a side note we point out that selection mechanisms in natural evolution as observed in life on Earth do in fact stem at some level from primary representation bias, so in some sense there is no secondary bias in natural evolution. This is not the case in artificial evolutionary systems that use explicit selection mechanisms. Fitness functions in ER are imposed at a very high level and cannot be reduced to be consistent with fundamental physical law.

C. Fitness Functions

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 task at hand. For optimization or classification applications, data sets and error minimization fitness functions can be applied and have proven to be very powerful. But these data set and error minimization functions are not generally applicable to ER or intelligent control learning because the information required to formulate them is equivalent to the information the system is intended to learn. If one had in hand a functional specification of a given intelligent control algorithm at the sensor/actuator level, it would make little sense to use this to train an agent. A control program could be written directly from the specification. For all but the most trivial intelligent autonomous control problems, an appropriate mapping between sensor inputs and actuator outputs is not known. Often only a description of the robot's final state in its environment is definable, hence this must be used to train the controllers. Standard methods of state space reinforcement learning (RL) such as Q-learning are also not applicable to non-trivial autonomous intelligent control problems

because adequate discrete state spaces cannot be formulated. There has been some work developing methods of converting continuous state spaces into discrete state spaces, and to applying RL to continuous high dimensional systems, but again, the information necessary to formulate these representations is in general equivalent to the information needed to specify the desired control algorithm.

In current research aimed at evolving populations of autonomous robot controllers capable of performing complex tasks, the fitness function is almost always the limiting factor in achievable controller quality. This limit is usually manifested by a plateau in fitness in later generations, and indicates that the fitness selection function is no longer able to detect fitness differences between individuals in the evolving population.

As mentioned above, for purposes of discussion, will define two fundamental classes of fitness functions used in ER. These are *behavioral* and *aggregate*. We will also define a class of fitness functions that represents a combination of these two basic ones, taking elements from each. These will be called *tailored* fitness functions and are representative of a general tendency of researchers in this area to create hand-formulated fitness functions that select for many features of control that they might believe will result in successful evolution of controllers capable of performing a desired task.

Behavioral fitness functions are task-specific hand-formulated functions that measure various aspects of what a robot is doing locally 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. Behavioral fitness functions measure simple action-response behaviors, low-level sensor-actuator mappings, or low-level actions the robot might perform. For example, if one wished to evolve robots to move about an environment and avoid obstacles, one might formulate a behavioral fitness function that includes 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 example robot controllers will evolve to produce particular actuator outputs in response to particular sensor inputs. 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.

It is evident that if one does not have a basic understanding of how to perform a particular task, a behavioral fitness function cannot be formulated. But because many, if not all, tasks studied in ER to date are very simple, researchers have been able to formulate behavioral fitness functions using their own expertise and intuitions. For this reason behavioral fitness functions have been used extensively up to this point in ER. Examples of the use of behavioral fitness functions can be found in [1][2][3].

Some behavioral fitness functions are selective for a desired control feature, rather than a precise sensor-to-actuator mapping. 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 many other formulations that could also produce the desired control feature. 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.

For the evolution of non-trivial behaviors, selection using behavioral fitness functions results mainly in the optimization of human-designed controller strategies, as opposed to the evolution or learning of novel intelligent behavior. Because the tasks studied so far in the field have been relatively simple, and in many ways aimed at general proof of concept research, the reliance on behavioral fitness functions has not been such an important issue. However, it is safe to say that the groundwork in ER has been laid. It is possible to train robot control systems to perform behavioral tasks that require them to operate autonomously in dynamic environments.

Aggregate fitness functions select only for high-level success or failure to complete a task. Selection is made without regard to how the task was actually 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. Evolution performed with an aggregate fitness function is sometimes called *all-in-one* evaluation. Until recently, aggregate fitness selection was largely dismissed by the ER community. This is because initial populations of controllers can be expected to have no detectable level of overall competence to perform non-trivial tasks (i.e. they are *sub-minimally competent*). Pure aggregate selection produces no selective pressure in sub-minimally competent populations at the beginning of evolution and hence the process cannot get started (the *bootstrap problem* [4]).

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.

Examples of aggregate fitness selection are found in [5][6][7].

Although there are many examples of the use of pure behavioral fitness functions, and some further examples of purely aggregating fitness functions, much of the ER research actually uses some form of hybrid between behavioral and aggregate selection. We will refer to these hybrid objective functions as *tailored fitness functions*.

Tailored fitness functions contain behavior-measuring terms, as well as aggregate terms that measure some degree or aspect of task completion that is divorced from any particular behavior. As an example, suppose a phototaxis behavior is to be evolved. A possible fitness function might contain one term that rewards the degree to which a robot turns toward the light source, and another term that rewards a robot that arrives at the light source by any means, regardless of the specific sensor-actuator behaviors used to perform the task.

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 during testing. This may seem at first glance to be free of *a priori* task solution knowledge or bias, 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 by the designer to the given problem, and often contain solution information implicitly or explicitly. Examples of work using tailored fitness functions can be found in [8][9][10].

To conclude our discussion of fitness functions we mention two variant methods of evolution that appear in ER. These are incremental evolution and competitive evolution.

Incremental evolution begins the evolutionary process by selecting for a simple ability upon which a more complex overall behavior can eventually 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. In many ways this can be thought of as a serialization of a complex behavioral or tailored fitness function. The initial fitness functions in incremental evolution can be tailored or behavioral, but the final applied fitness function might be purely aggregate.

Competitive evolution (competitive 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.

A variant upon competitive evolution is co-competitive evolution in which 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 [11][12][13]. 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. The research presented in [11][12][13] shows this effect to a degree, but results from other areas of evolutionary computing suggest that given the correct evolutionary conditions, pure aggregate selection combined

with intra-population competition can result in the evolution of very competent systems [14][15].

II. EVOLUTIONARY ROBOTICS SO FAR

The field of ER has been reviewed in several publications [16][17][18]. Much of the research focuses on evolving controllers for simple tasks such as phototaxis [19][20], object avoidance [21][22], simple forms of navigation [23][24], or low-level actuator control for locomotion [25][3].

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 discussed below use some form of evolutionary algorithm roughly equivalent to that which was outlined in the introduction to this chapter. It is true that some algorithms may show a two-fold (or even ten-fold) increase in training efficiency over others, but so long as the search space and controller representation space are not over-constrained it is the fitness function that finally determines the achievable performance.

We consider mainly evolutionary robotics work that has been 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 [50][51]. 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 representations or rely on sensors that report unobtainable data or conceptual data. That said, learning in simulation with transfer to real robots has been repeatedly demonstrated to be viable over the last decade [6][26][27][28]. Much of this work has involved new physics- and sensor-based simulators. 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.

Early research efforts that might be considered precursors to evolutionary robotics done in the late 1980's consisted of learning simple navigation abilities in purely simulated agents [50][52]. In the first decade of evolutionary robotics work (1990-2000), the field moved from an almost non-existent state to become a field of research in which numerous projects produced real robots relying entirely on evolved controllers for interaction with their environments.

Locomotion in combination with obstacle avoidance in legged robots has been reported in several studies [29][3][21][5]. Filliat et al. [29] evolved locomotion and object avoidance controllers for a hexapod robot using neural networks composed of threshold neurons. Controllers were evolved in simulation and transferred to real robots for testing. Jakobi et al. [3] described the use of minimal

simulation to evolve controllers for an eight-legged robot with sixteen leg actuators. Kodjabachian et al. [21] describe the incremental evolution of walking, object avoidance and chemotaxis in a simulated six-legged insectoid robot. Hornby et al. [5] describe the evolution of ball chasing using an 18-DOF quadruped robot.

Peg pushing behaviors were evolved in [30][4]. This task required robots to push small cylinders toward a light source. In [31] Lee et al. investigated a similar box-pushing behavior using Genetic Programming (GP).

Several examples of competition in the form of co-evolution of competing species have been reported in the literature. Cliff and Miller investigated the co-evolution of competing populations of predator and prey robots [32][13]. Similar works have been reported in [17][11][12].

Evolution of controllers using competition within a single population (intra-population competition) is investigated in [28].

The most complex tasks addressed in the literature involve some form of sequential action. Nolfi [33] reports on the evolution of a garbage collection behavior in which a robot must pick up pegs in an arena and deposit them outside the arena. Ziemke [34] studied the evolution of robot controllers for a task in which a robot must collide with objects ("collect" them) in one zone and avoid them in another. In [35] Floreano et al. report on the evolution of a behavior in which robots move to a light and then back to a home zone. Another example of evolving controllers for a relatively complex task is reported in Tuci et al. [36]. Robot controllers evolved to produce lifetime learning in order to predict the location of a goal object based on the position of a light source.

Flocking behaviors have also been investigated. Ashiru describes the evolution of a simple robot flocking behavior in [37]. A robot coordination task in which two robots evolve to move while maintaining mutual proximity is reported by Quinn in [38]. Baldassarre et al. [39] evolved homogeneous controllers for a task in which four robots must move together in a small group toward a light or sound source. In [40] aggregation of small robots into a larger structure is investigated and makes use of a relatively complex hand-formulated fitness function.

In the early 2000's evolution of body and mind (morphology and controller) was achieved using the innovation of elemental modular components that were both amenable to simulation, and relatively easy to fabricate [6][26]. It must be noted that these works have not produced significant advances in controller complexity *per se*, but rather they have shown that evolution in simulated environments can indeed be used to produce physically viable robot minds and bodies.

It may still be the case that certain environments are beyond the ability of modern methods to simulate for the purpose of evolutionary learning, but this cannot now be considered to be the stumbling block that it once was. Numerous experiments and systems have shown that the intuitively compelling arguments supporting embodiment as a requirement for low-level learning entertained by researchers in the late 1980's and early 1990's [51] are in fact not correct [6][26][27][28]. The universe is by no means its own best simulation [50].

Recent years have also seen a significant methodological change related to fitness function usage in the field. Aggregate fitness functions have been used to reproduce many of the results first obtained using more complicated hand-formulated fitness functions. Some earlier works, especially in the area of incremental evolution, made the claim that the more complex forms of fitness evaluation were necessary to achieve successful evolution of the behaviors studied. This however has been shown not to be the case.

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 [19][11][22][20]. Also, there have been numerous examples of successful evolution of controllers in simulation with transfer to real robots [1][33] [41][6][5][42][43].

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 and fitness selection remain largely unresolved. The majority of ER research presented in the literature employs some form of hand-formulated, task-specific fitness selection function that more or less defines how to achieve the intended task or behavior. The most complex evolved behaviors to date consist of no more than three or four coordinated fundamental sub-behaviors [33][35][28][27]. In [33], the fitness selection method used was relatively selective for an *a priori* known or pre-defined solution. In [35][28][27] 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, which in some cases used complex fitness functions that heavily biased the evolved controllers toward an *a priori* known solution.

III. Evolutionary Robotics and Aggregate Fitness

In this section we focus on evolutionary robotics research that has used aggregate fitness functions.

Several robot actuator control tasks have been investigated using aggregate or near-aggregate fitness functions. These include gait evolution in legged robots [44][7][45][46], and flying lift generation in a flying robot [25]. Simple actuator coordination tasks do not fall under the heading of environmentally situated intelligent autonomous robot control and do not usually produce complex

reactions to environmental sensor stimuli. However, they do involve the application of evolutionary computing methods to evolve novel control, and are included in this section along with the other evolved autonomous controller research.

In [5] the evolution of a ball-pushing behavior using an 18-DOF quadruped robot (Sony AIBO) is described. The fitness function used can be considered to be aggregate. The function measures the degree of success of moving the ball simply by measuring the total distance that the ball was moved over the course of an evaluation trial.

In [44] the authors use embodied evolution to develop gaits for a hexapod robot. An aggregate fitness function was used that measured the distance traveled by the robot while walking on a treadmill.

Both [6] and [26] described separate examples of systems in which whole robots (bodied 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 purely aggregate fitness function and contained no other features of potential control solutions or of possible robot morphologies.

[7] reported on the embodied evolution of a locomotion behavior in a robot relying on tactile sensors for object detection. The fitness function simply measured the arc-length distance traveled by the robot (as reported by a swivel wheel and odometer attached to the robot) over a given evaluation period:

In [25] embodied evolution was used to develop lift-generating motions in a winged flying robot. A near aggregate fitness function was used that measured height obtained by the robot at each time step.

In [47] 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. An essentially aggregate fitness function was used that averaged magnitude of velocity over each trial period.

[48] described the evolution of morphology and control for 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. A near-aggregate fitness function (the same as that used in [6] and [26]) was used that measured total net locomotion distance over the course of a trial period. 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.

[45] presents another example of 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 [44]. Distance for the aggregate fitness function was determined using images taken from an overhead camera.

Gait learning using a slightly different aggregate fitness function is given in [46]. In this paper, a Sony AIBO quadruped robot was used, and again, the evolutionary process was embodied in the real robot. Here fitness was measured as average speed achieved by the robot.

IV. MAKING AGGREGATE SELECTION WORK

This section discusses methods for overcoming difficulties associated with using aggregate fitness functions to evolve populations of mobile robot controllers for specific tasks.

The foremost problem with aggregate selection is that it lacks selective power early in evolution. One way to address this is to use a behavioral or 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 purely aggregate success/failure selection alone in the later part of evolution, thus relaxing biases introduced by the initial fitness function. The term *bootstrap mode* is used to refer to such an initial training function.

It is not clear what legacy in the evolving population might be left by biases introduced by an initial bootstrap mode fitness function. Once started down one path, innovation might be inhibited even if the initial biases are removed. Also, designers still need to have some idea what kinds of fundamental abilities are needed to allow the evolving controllers to have some chance at completing the task. Even with these problems, the use of an initial bootstrap mode in conjunction with later pure aggregate selection is a viable method for some non-trivial tasks and will likely be studied a great deal in the coming years.

Using intra-population competition in conjunction with aggregate selection may also improve the quality of evolved controllers, at least for tasks that are inherently competitive. 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 hadn't been obstructed by the first robot.

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.

V. AGGREGATE SELECTION AND COMPETITION

In this section we provide an overview of our own ER research [28][42][28]. The specific aim of the experiments reviewed here was to extend aggregate selection using the methods discussed in the previous section.

We used aggregate selection with a minimal bootstrap mode in conjunction with direct intra-population competition. The bootstrap mode was triggered only

when no controller in the current generation of the evolving population showed any detectable ability to complete the overall task.

As in the majority of ER research, we used neural networks to control our robots. Populations of neural network based controllers were evolved to play a robot version of the competitive team game *Capture the Flag*. In this game, there are two teams of mobile robots and two stationary goal objects. All robots on one team and one of the goals are of one color (red). The other team members and their goal are another color (green). In the game, robots of each team must try to approach the other team's goal object while protecting their own goal. The robot which first comes within a range of its opponent's goal wins the game for its team. Winning the game here is the task the robots learn to solve. The game is played in maze worlds of varying configurations.

The competitive task, in the form of a game, allows for both simple and complex strategies to arise. Although the simplest solution to this game task that might succeed is to wander about the environment until the opponent's goal is found, more complex strategies are more efficient. Such a simple strategy will usually fail against a more competent opponent. The competitive element of the evolutionary environment is needed to drive controllers toward better solutions, and this is where our research differs from most other research in the field. We employ direct competition during evaluation between robots. This means that the actions of one robot can alter the fitness evaluation of another directly. This also produces a changing fitness landscape over the course of evolution (the Red Queen Effect [32]).

The neural networks are essentially blank slates at the beginning of evolution. They are made up of randomly interconnected neurons with weighted connections initialized using values from a random distribution and they contain no information related to the task to be learned. In addition to learning any game-specific behaviors, the robots must also learn all aspects of locomotion and navigation. The robot controllers relied solely on processed video inputs for sensing their environment, and the best-performing neural network controllers contained on the order of 100 neurons and 5000 connections. Because of the very large number of sensor inputs, and the number of different types of objects the robots needed to learn to recognize (or at least respond to), the actual task learned by the controller is in some ways much more difficult than those studied in other related work.

The genetic representation of the neural controllers was a direct encoding of the network weights and connection topology. This consisted of a matrix of real-valued elements in which each element value represented a connection weight, and the location of each element represented the position of that connection in the overall network. The matrix also contained several additional columns of formatted fields that specified neuron types and time delays.

During evolution, only mutation was used. Network weights, connectivity and network topology were evolved. For the work discussed below, populations of 40 individual controllers were evolved. The evolutionary conditions and parameters are summarized in table 1 below.

Table 1. Environmental and experimental parameters used during evolution.

Parameter	Setting
Population size	40
Sensor inputs	150
Initial network size	60 neurons
Chance of adding or removing a neuron (during mutation)	70%
Weight initialization range	[-1 1], uniform distribution
Weight mutation magnitude	[-1 1], uniform distribution
Weight mutation rate	25%
Initial feed forward connectivity	60%
Initial feedback connectivity	20%
Chance of adding or removing a connection (during mutation)	70%
Elitism level (per generation)	Single best from previous generation
Population replacement rate	50%
Generations (per evolution)	650

The physical robots used in this work were the EvBots [37][29]. The robots were fully autonomous and performed all vision processing and control computation on board. Figure 1 shows a photograph of two EvBots. Each robot has been fitted with a colored shell. The shells were used in the *Capture the Flag* game behavior and served to differentiate robots on different teams.

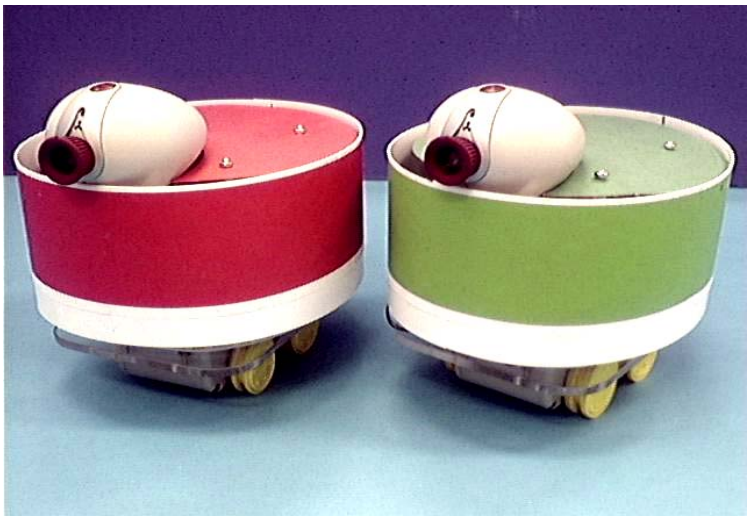

Figure 1. EvBot robots fitted with colored shells.

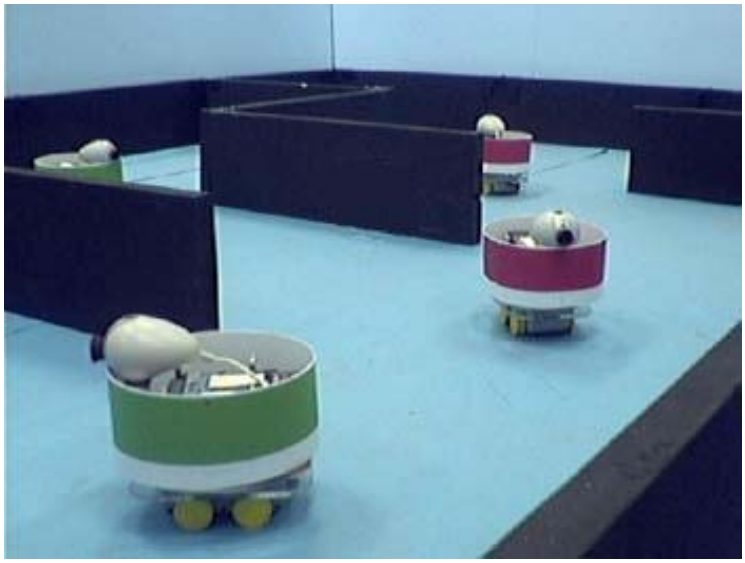

Figure 2. Robots in the CRIM testing maze.

Testing of the robot controllers was performed in the reconfigurable CRIM robot maze at the Center for Robots and Intelligent Machines at North Carolina State University (Figure 2). The controllers themselves were evolved in a simulation environment coupled to the real robots and maze environment at the sensor and actuator level [42].

Fitness for individual controllers was based on their performance in competition in tournaments of games. During each generation, a single tournament of games was played. A bimodal training fitness selection function was used. The fitness function has an initial bootstrap mode that accommodates sub-minimally competent seed populations and a second mode that selects for aggregate fitness based only on overall success or failure (winning or losing games). The triggering of modes was decoupled from any explicit aspect of the training environment or specific generation number and was related only to the current behavior of the population.

The fitness function was applied in a relatively competitive form in which controllers in the evolving population competed against one another to complete their task—to win the game. In a given generation, if any robot controller was able to complete the task, then all information from the bootstrap mode was discarded for all the robots regardless of their individual performances and selection for the whole population was conducted with purely win/lose information.

Formally, fitness $F(p)$ of an individual p in the population **P** at each generation was calculated by:

$$F(p) = F_{\text{mode\_1}}(p) \oplus F_{\text{mode\_2}}(p) \tag{1}$$

where $F_{\text{mode\_1}}$ is the initial minimal-competence bootstrap mode and $F_{\text{mode\_2}}$ is the purely aggregate success/failure-based mode. Here $\oplus$ indicates exclusive-or, dependent on $F_{\text{mode\_2}}$: if the aggregate mode's value is non-zero, it is used and any value from $F_{\text{mode\_1}}$ is discarded. Otherwise fitness is based on the output of $F_{\text{mode\_1}}$. $F_{\text{mode\_1}}$ is formulated to return negative values and returns 0 when maximized or if $F_{\text{mode\_2}}$ is active. $F_{\text{mode\_2}}$ in contrast returns positive values based on number of game-wins, if any, and this allows for a simple mechanism to track which mode is active at any generation over the course of evolution.

The minimal competence bootstrap mode selects for the ability to travel a distance D through the competition maze environment. The general form of mode 1 is as follows:

$$F_{\text{mode\_1}} = F_{dist} - s - m \tag{2}$$

where F_{dist} calculates a penalty proportional to the difference between distance d traveled by the best robot on a team, and the minimal competence distance D, which is defined as half the length of the training environment's greatest dimension. In Equation (2), s and m are penalty constants applied when robots on a team become immobilized or stuck (by any means), or when controllers produce actuator output commands that exceed the range of the actuators (the wheel motors) respectively.

The rationale for this particular bootstrap mode is that if a robot can navigate at least partway through a given environment without becoming ensnared on walls, other robots, or other obstacles, then it has some chance of running across its quarry (i.e. the opponent goal).

The second mode of the fitness function $F_{\text{mode\_2}}$ is classified as aggregate because it produces fitness based only on success or failure of the controllers to complete the overall task (i.e. winning the game by finding and touching the opponent's goal first). The formulation of the success/failure mode of the fitness function is determined by the competitive nature of the training algorithm. In each generation, a tournament of games involving all the individuals in the population was conducted. Each individual played two games against another member of the population (the opponent). Note that the opponent was selected at random from the previous generation of the population at the beginning of each tournament and all controllers competed against that one robot. The reason for this was to reduce the stochastic differences during evaluation introduced by the environment and random opponent selection. The possible outcomes of these games incurred different levels of fitness and are summarized in Table 2 below.

Table 2. Fitness points awarded by the aggregate success/failure mode F_{mode_2}, for pairs of reciprocal games during a generational tournament.

Game Pair Outcomes	Fitness Points Awarded
win-win	3
win-draw	1
win-lose	.5

Note that in cases where no win occurs during the entire tournament F_{mode_1} is used to determine negative fitness values.

Over the course of a typical controller evolution run, the fitness function started using purely bootstrap mode, progressed to using a mixture of bootstrap and aggregate, and then in the later parts of evolution relied exclusively on pure aggregate selection. Figure 3 shows fitness values and number of games won per generation over the course of an evolutionary run. The active mode of the fitness function used during each generation is indicated at the top of the figure. No controller in the population was able to win a game before the 60th generation of training. With a population size of 40 (a typical size for this research) this would represent about 4000 games and indicates that the initial and early forms of the population have virtually no ability to complete the overall task and that use of the aggregate fitness mode alone would indeed fail. Between the 60th and 160th generations the fitness function oscillates between the two modes, relying more heavily on the pure aggregate mode as evolution continues. After the 160th generation the bootstrap mode is not invoked again, and selection is based completely on aggregate selection. Note that the differing fitness values correspond to the fitness values listed in Table 2 when the aggregate mode is active. In addition, note that the incremental climb in fitness in the first 60 generations when fitness is dominated by the bootstrap mode is typical of ER trainings in which a behavioral fitness function is used. Further, as the aggregate fitness mode becomes dominant, absolute fitness becomes less of an indicator of population refinement. Since the high score is 3 points, any generation that has a controller that can win two games it plays will have this as the best fitness.

Over the course of evolution the robots learned how to navigate and locate goals. They could differentiate between the different types of objects in their environment as judged by their different responses to these. The robot controller also evolved some temporal behaviors, but relied mainly on reactive control.

The results shown in Figure 3 summarize data collected over a particular evolutionary training session. A series of such evolutions were performed, and produced generally similar results, but due to very long simulation times (on the order of three weeks of computation time on a 3.2 GHz X86 class dual processor machine) no statistically significant data summarizing a putative general case were generated.

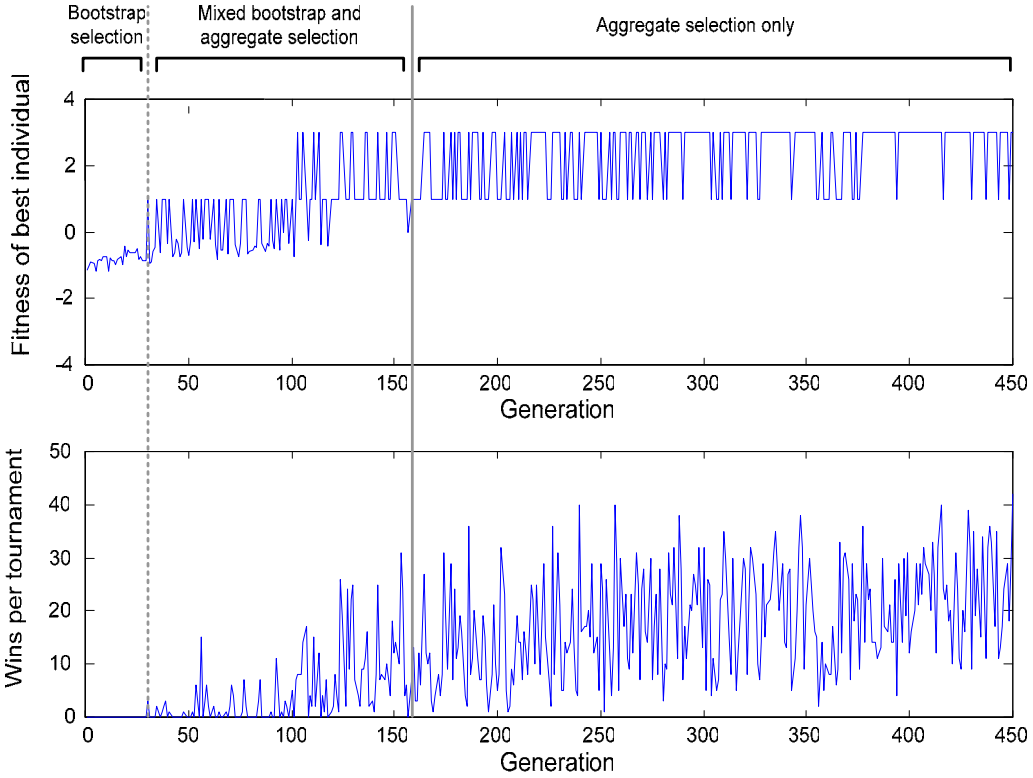

Figure 3. Fitness of best individual and number of wins per generation are shown over the course of evolution for an evolving population. Modes of fitness function operation are also indicated.

Example games played with evolved controllers are shown in Figure 4. The first panel shows a screen shot of the simulation environment while the second panel shows a view of the physical maze environment and the real robots taken with an overhead video camera. In both of the example games the robots are shown in their final positions at the end of the games. The paths taken by the robots are indicated by lines superimposed on the images.

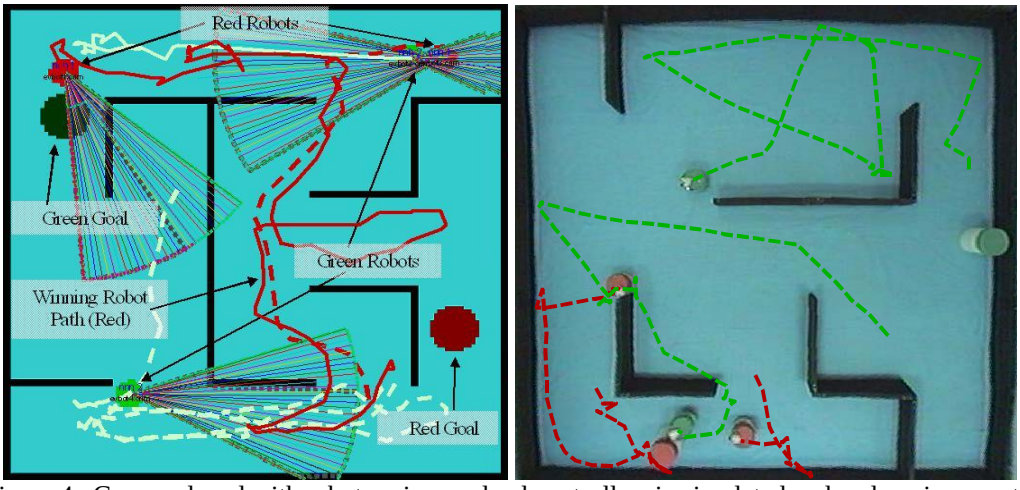

Figure 4. Games played with robots using evolved controllers in simulated and real environments.

The game sequences of Figure 4 demonstrate that controllers have evolved some degree game-playing behavior. Because a relatively competitive fitness selection metric was used to drive the evolutionary process, the absolute quality of the robot controllers is not known. To address this, after the evolutionary training process was complete, the final evolved controllers were tested against a hand-coded knowledge-based controller of well-defined abilities.

The hand-coded controller was not used in any way during the training of the neural network-based controllers. As far as the evolved controllers were concerned, the rule-based controller was a novel controller not seen during training. In addition, the post-training games were played in a maze configuration not used during training in order to rule out the possibility that the controllers memorized a particular maze topology as part of their learned strategies.

In order to obtain a reliable result a series of 240 games between the rule-based controller and the best trained neural network controller was conducted in an environment similar to the ones shown in Figure 4. Each game during the tournament was initialized with a new randomly generated set of starting positions for robots and goals. Figure 5 shows the results of this tournament. The best-evolved neural controller won 108 games, the knowledge-based controller won 103 games, and 29 games were played to a draw.

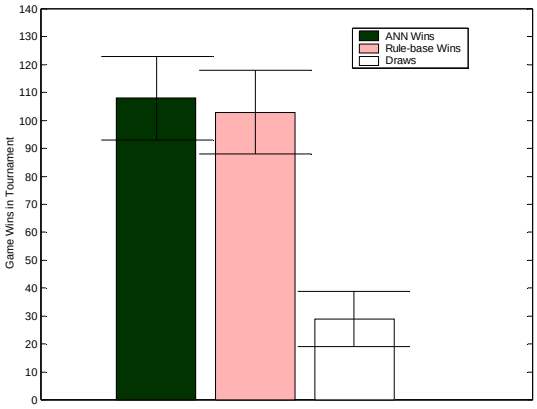

Figure 5. Bar graphs displaying evolved controller and knowledge-based controller competition data collected during a tournament of 240 randomly initialized games. Data are shown with 95% confidence intervals.

The data do not show that the evolved controllers were significantly better than the hand-coded controller. Even with the large number of games played, the total number of wins for each type of controller overlapped within 95% confidence intervals. The evolved controllers played competitively with the hand-written controllers, though, and this result is statistically significant.

In marked contrast to the knowledge-based controllers, the neural network-based controllers displayed complex trajectories that were extremely difficult to predict exactly. Although it may be possible to qualitatively analyze the evolved controller behaviors to a degree, such analysis is not at all necessary to the evolutionary process. Competition alone during evolution was responsible for

driving the complexity of the controllers to a level at which they could compete the hand-coded controller. Human knowledge was required to formulate the minimal competence mode of the fitness function, but this only selected for minimal navigation abilities.

The work discussed above used a bootstrap selection mode early in evolution before transitioning to an aggregate mode. This raises the question of how these experiments compare to similar work that used only purely aggregate selection from the beginning. For comparison, we attempted to use purely aggregate selection without the initial bootstrap mode. For our particular competitive game, purely aggregate selection was not able to drive the evolution of fit controllers (data not shown). However, in [49], carefully structured environmental-incremental evolution, in which the environment used during training was made progressively more difficult, did produce competent controllers in a simulated experiment. The task in [49] required robots to find and pick up objects in an arena, and then deposit them outside the arena. The majority of the other evolutionary robotics work reported in the literature that used purely aggregate fitness functions evolved simple locomotion behaviors and is not directly comparable to more complex tasks.

In [14] networks for playing Checkers were evolved to the near-expert level (able to beat 95% of human players) using a purely aggregate selection scheme. This was a non-robotic task, and the game of Checkers lacks the dynamic sensor-motor-environment feedback loops characteristic of autonomous robotic systems, but the work demonstrated that it is possible to evolve very competent systems for difficult tasks.

VI. CONCLUSION

We conclude the chapter with a discussion of the long-term prospects of using artificial evolution to generate highly complex autonomous mobile robot controllers. In spite of recent advances in ER, there remain fundamental unsolved problems in the field that prevent its application to many non-trivial real-world problems.

Although the field of evolutionary robotics has expanded and developed greatly in the past decade, in some sense no substantive improvement in overall controller sophistication has been achieved during this time. This is in spite of continued advances in computer speeds, availability of robot platforms, and efficiency of algorithms.

We hypothesize that the lack of improvement in the field of ER is fundamentally a result of an inability to formulate effective fitness functions for non-trivial tasks. For complex tasks, behavioral fitness functions must contain more and more features of a known solution. In effect, the fitness function becomes a recipe for evolving a solution containing mainly *a priori* known features and more or less defines all of the major features of the evolved solutions.

We believe that aggregate selection must be used in one form or another to achieve complex behaviors that are not known by researchers and developers *a priori*, and this is crucial for the application of ER to sophisticated robot learning in dynamic uncharacterized environments. This is not just a problem for ER, but affects any form of learning in uncharacterized environments that require some form of feedback to drive the learning process.

The work in [14] and also in [15] use aggregate selection to generate very competent systems for playing the games of Checkers and Go respectively. However, the current robotics work has not reached these levels of proficiency. Evolution conducted with one or more carefully formulated bootstrap modes, followed by aggregate selection performed in a fidelity training environment with adequate computing power will very probably be able to generate controllers an order of magnitude more complex than the current standard fitness evaluation methods in ER. But this is in no way a general solution to autonomous control learning. Controllers created using most current ER methods might be capable of performing one or two distinct elemental tasks, and the most sophisticated controllers evolved perform no more than five or six coordinated elemental tasks. An order of magnitude improvement would not represent a significant improvement over current non-learning-based autonomous robot controllers, although it would allow ER methods to at least be competitive with other state-of-the-art autonomous robot control design methods.

Intra-population competition during fitness evaluation, in combination with aggregate or near-aggregate selection, can produce controllers of significant complexity, but many if not most tasks of interest cannot be reduced to a purely competitive form in which one robot's behavior directly impinges upon the behavior (and thus the fitness) of another co-evolving robot. This direct intra-population competition within the robot's testing environment is essential in order to drive evolution to produce sophisticated agents, at least if it is applied as in the experiments discussed in this chapter. So again, we do not yet have a tool that can provide generalized learning for most non-trivial problems.

A second and more esoteric problem with aggregate selection, as well as other selection mechanisms in artificial evolving systems, is that they impose a particularly high-level of task bias on the evolving agents in a way that cannot be related to natural evolutionary processes. This is not to say that every aspect of natural evolution must be duplicated in artificial evolution systems, but because the natural development of life on Earth appears to be our only example of truly complex agents arising from lifeless origins, it is unclear which aspects of natural evolution are essential, and which are merely artifacts of life as it developed on Earth.

Artificial evolution when applied to autonomous robots attempts to evolve abilities to perform specific tasks. In natural systems, there is no particular fundamental bias toward any particular functionality. This is an important and perhaps subtle difference between natural and artificial evolution. Natural evolution does not evolve creatures with specific abilities *per se*. Nature used (uses) exactly the same fundamental driving selection force to evolve, for example, trees as it did to evolve birds. The underlying bias toward particular

functionality (i.e. photosynthesizing or flying) in natural systems is very low. Further, all forms of bias seen in nature are reducible to fundamental physical law (even if this reduction is beyond our abilities to elucidate). Because representation (i.e. the physical universe) and selection pressure in nature are consistent, fundamental bias is very low, yet complexity can be represented. It is possible that in order for anything approaching life-like abilities to arise, all forms of selection bias must be kept at extremely low levels, and must be consistent with fundamental representation. This places into question whether it is possible to evolve agents for particular predefined tasks in general.

There are no artificial evolutionary systems in which selection and representation are fully consistent. Currently, all artificial life systems, including all evolutionary robotics systems, that have shown any signs of developing autonomous capabilities use some form of fitness evolution that is imposed on the evolving systems. Self-organizing systems such as cellular automata have been studied, and these perhaps do not suffer from secondary biases, but no such system has been shown to have the potential to generate intelligent interactive behaviors that are necessary for autonomous robotic systems that are intended to operate in dynamic environments.

Acknowledgment: The authors would like to thank Brenae Bailey for editorial assistance and insightful input related to this work.

References

[1] H.H. Lund, O. Miglino, "From simulated to real robots," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 362-365.

[2] 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.

[3] N. Jakobi, "Running across the reality gap: Octopod locomotion evolved in a minimal simulation," in *Evolutionary Robotics: First European Workshop, EvoRobot98*, P. Husbands, J.A. Meyer, Eds., Springer-Verlag, 1998, pp. 39-58.

[4] K. Kawai, A. Ishiguro, P. Eggenberger, "Incremental evolution of neurocontrollers 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.

[5] 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 *Evolvable Systems: from Biology to Hardware; Proceedings of the Third International Conference (ICES 2000)*, J. Miller, Ed., *Lecture Notes in Computer Science*, vol. 1801, Springer, 2000, pp. 80-89.

[6]H. Lipson, J.B. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, no. 6799, pp. 974-978, Aug. 31, 2000.

[7] 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)*, Sept. 10-24, 2001.

[8] A.C. Schultz, J.J. Grefenstette, W. Adams, "RoboShepherd: learning a complex behavior," in *Robotics and Manufacturing: Recent Trends in Research and Applications*, vol. 6, pp. 763-768, 1996.

[9] D. Keymeulen, M. Iwata, Y. Kuniyoshi, T. Higuchi, "Online evolution for a self-adapting robotic navigation system using evolvable hardware," *Artificial Life*, vol. 4, no. 4, pp. 359-393, 1998.

[10] 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)*, Aug. 14-16, 2002, HP Bristol Labs, U.K.

[11] S. Nolfi, D. Floreano, "Co-evolving predator and prey robots: Do 'arms races' arise in artificial evolution?" *Artificial Life*, vol. 4, no. 4, pp. 311-335, 1998.

[12] G. Buason, N. Bergfeldt, T. Ziemke, "Brains, bodies, and beyond: competitive co-evolution of robot controllers, morphologies and environments," *Genetic Programming and Evolvable Machines*, vol. 6, no. 1, pp. 25–51, 2005.

[13] D. Cliff, G.F. Miller "Co-evolution of pursuit and evasion II: simulation methods and results," in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB96)*, P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, S.W. Wilson eds., MIT Press, Bradford Books, 1996, pp. 506-515.

[14] K. Chellapilla, D.B. Fogel, "Evolving an expert checkers playing program without using human expertise," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 422-428, 2001.

[15] 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.

[16] S. Nolfi, D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines, The MIT Press, Cambridge Massachusetts, 2000.

[17] I. Harvey, P. Husbands, D. Cliff, A. Thompson, N. Jakobi, "Evolutionary robotics: the Sussex approach," *Robotics and Autonomous Systems*, vol. 20, no. 2-4, pp. 205-224, 1997.

[18] L.A. Meeden, D. Kumar, "Trends in evolutionary robotics," in *Soft Computing for Intelligent Robotic Systems*, L.C. Jain, T. Fukuda, Eds., Physica-Verlag, New York, NY, 1998, pp. 215-233.

[19] 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.

[20] R.A. Watson, S.G. Ficici, J.B. Pollack, "Embodied evolution: distributing an evolutionary algorithm in a population of robots," *Robotics and Autonomous Systems*, vol. 39, no. 1, pp. 1-18, Apr. 2002.

[21] J. Kodjabachian, J.-A. Meyer, "Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle avoidance in artificial insects," IEEE Transaction on Neural Networks, vol. 9, no. 5, pp. 796-812, Sept. 1998.

[22] D. Floreano, F. Mondada , "Evolution of homing navigation in a real mobile robot," *IEEE Transactions on Systems, Man, Cybernetics Part B: Cybernetics*, vol. 26, no. 3, pp. 396-407, 1996.

[23] R.D. Beer, J.C. Gallagher, "Evolving dynamical neural networks for adaptive behavior," *Adaptive Behavior*, vol. 1, no. 1, pp. 91-122, 1992.

[24] J. Grefenstette, A. Schultz, "An evolutionary approach to learning in robots," Machine Learning Workshop on Robot Learning, New Brunswick, 1994.

[25] 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, July 9-13 2002, Morgan Kaufmann, pp. 1279-1285.

[26] 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.

[27] G. Capi, K. Doya, "Evolution of recurrent neural controllers using an extended parallel genetic algorithm," *Robotics and Autonomous Systems*, vol. 52, no. 2-3, pp. 148-159, 31 Aug. 2005.

[28] A.L. Nelson, E. Grant, G.J. Barlow, M. White, "Evolution of autonomous robot behaviors using relative competitive fitness," in *Proceedings of the 2003 IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS'03) Modeling, Exploration, and Engineering Systems*, Boston MA, Oct. 1-3, 2003, pp. 145-150.

[29] D. Filliat, J. Kodjabachian, J.A. Meyer, "Incremental evolution of neural controllers for navigation in a 6 1egged robot," in Sugisaka and Tanaka, Eds., *Proc. Fourth International Symposium on Artificial Life and Robotics*, Oita Univ. Press, 1999.

[30] 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 *Proceedins of the 1999 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, 1999, pp. 239-244.

[31] 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 IEEE 3rd International Conference on Evolutionary Computation*, May 20-22, 1996, pp. 384-389.

[32] D. Cliff, G.F. Miller, "Tracking the red queen: measurements of adaptive progress in co-evolutionary simulations," in *Proceedings of the Third European Conference on Artificial Life: Advances in Artificial Life (ECAL95)*. F. Moran, A. Moreno, J.J. Merelo, P. Cachon eds., Lecture Notes in Artificial Intelligence 929, Springer-Verlag, 1995, pp. 200-218.

[33] S. Nolfi, "Evolving non-trivial behaviors on real robots," *Robotics and Autonomous Systems*, vol. 22, no. 3-4 pp. 187-198, 1997.

[34] T. Ziemke, "Remembering how to behave: Recurrent neural networks for adaptive robot behavior," in Medsker and Jain Eds., *Recurrent Neural Networks: Design and Applications*, Boca Raton, CRC Press, 1999.

[35] D. Floreano, J. Urzelai, "Evolutionary robots with on-line self-organization and behavioral fitness," *Neural Networks*, vol. 13, no. 4-5, pp. 431-443, June 2000.

[36] E. Tuci, M. Quinn, I. Harvey, "Evolving fixed-weight networks for learning robots," in *Proceedings of the 2002 Congress on Evolutionary Computing*, Honolulu HI, vol 2, 2002, pp 1970-1975.

[37] I. Ashiru and 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.

[38] 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.

[39] 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, Sept. 8-13, 2002, pp. 11-22.

[41] P. Nordin, W. Banzhaf, M. Brameier, "Evolution of a world model for a miniature robot using genetic programming." *Robotics and Autonomous Systems*, vol. 25, no. 1-2, pp. 105-116, 31 Oct. 1998.

[42] 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, Oct. 27-31, 2003, pp 2273-2278.

[43] 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.

[44] 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, April 17-19 2000.

[45] V. Zykov, J. Bongard, H. Lipson, "Evolving dynamic gaits on a physical robot," *2004 Genetic and Evolutionary Computation Conference (GECCO)*, Seattle, WA., 2004.

[46] 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.

[47] J. Zufferey, D. Floreano, M. van Leeuwen, T. Merenda "Evolving vision based flying robot," in *Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision*, eds. Bülthoff, Lee, Poggio, Wallraven, LNCS 2525, pp. 592-600, Berlin, Springer-Verlag.

[48] 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, Mass, Sept.12-15, the MIT press, 2004, pp. 94-99.

[49] 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.

[50] R.A. Brooks, "Artificial life and real robots," in Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, F.J. Varela, P. Bourgine, eds, MIT Press/Bradford Books, Cambridge, MA, 1992, pp. 3-10.

[51] R.A. Brooks, "Elephants don't play chess," *Robotics and Autonomous Systems*, vol. 6, pp. 3-15, 1990.

[52] J.R. Koza, "Evolution of subsumption using genetic programming," in *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, F.J. Varela, P. Bourgine Eds., pp. 110-119, The MIT Press, Cambridge, MA, 1992.

[53] 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.