A.L. Nelson, E. Grant, "Using direct competition to select for competent controllers in evolutionary robotics," Robotics and Autonomous Systems, vol. 54, no. 10, pp. 840-857, Oct. 2006.

Using Direct Competition to Select for Competent Controllers in Evolutionary Robotics

Andrew L. Nelson and Edward Grant

Abstract--Evolutionary Robotics (ER) is a field of research that applies artificial evolution toward the automatic design and synthesis of intelligent robot controllers. The preceding decade saw numerous advances in Evolutionary Robotics hardware and software systems. However, the sophistication of resulting robot controllers has remained nearly static over this period of time. Here, we make the case that current methods of controller fitness evaluation are primary factors limiting the further development of ER. To address this, we define a form of fitness evaluation that relies on intra-population competition. In this research, complex neural networks were trained to control robots playing a competitive team game. To limit the amount of human bias or know-how injected into the evolving controllers, selection was based on whether controllers won or lost games. The robots relied on video sensing of their environment, and the neural networks required on the order of 150 inputs. This represents an order of magnitude increase in sensor complexity compared to other research in this field. Evolved controllers were tested extensively in real fully-autonomous robots and in simulation. Results and experiments are presented to characterize the training process and the acquisition of controller competency under different evolutionary conditions.

Index Terms--Evolutionary robotics; evolutionary neural networks; swarm robotics; developmental robotics.

I. INTRODUCTION

Evolutionary robotics (ER) is an emerging area of research within the more general field of autonomous robot control. The primary goal of evolutionary robotics is to develop automatic methods for synthesizing intelligent autonomous mobile robot controllers. These methods should not require hand coding or in depth human knowledge of the particular control tasks for which the controllers are intended.

Typically ER applies population-based artificial evolution to evolve autonomous robot controllers. The process of controller evolution consists of repeating cycles of controller fitness evaluation and selection that are roughly analogous to a generation in natural evolution. During each cycle, or generation, individual controllers taken from a larger population of controllers perform a task or engage in 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. Following this, each controller's performance is evaluated based on a fitness selection function (objective function). In the final step of every cycle, a genetic algorithm (GA) is applied. The GA uses information generated by the fitness selection function to select and propagate the fittest individuals in the current population to the next generation population. During propagation, controllers are altered slightly 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.

Possibly the most important unanswered question within the field of ER is whether the methods used so far to obtain simple proof-of-concept results can be generalized to produce more sophisticated autonomous robot control systems. In turn, a key issue related to the successful evolution of autonomous robot controllers is the specification of a fitness selection function.

A. Research Goals

The development of methods for general fitness selection during evolution of controllers is crucial to the future of ER. This view is reflected in some recent literature [13] and has been noted previously in [11]. As early as the mid 1960's it was pointed out that creating a method of fitness selection capable of selecting for complex behavior was likely to be difficult [15].

One of the main goals of the research presented in this paper is to investigate the application of aggregate success/failure selection in combination with direct intra-population competition to evolve complex neural networks using numerous processed video sensor inputs to perform a non-trivial autonomous control task. In order to address the issue of initial populations having no detectable level of fitness, we also introduce the concept of multi-modal fitness selection. This is discussed in detail in Section V.

In this work, populations of neural network-based robot 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 the first 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 in contact with its opponent's goal wins the game for its team. The game is played in maze worlds of varying configurations.

The evolved controllers are tested in competitions of 240 games against hand-coded knowledge-based controllers. Results show that evolved controllers are competitive with the knowledge-based controllers and can win a modest majority of games in a large tournament in a challenging maze world configuration. This work extends research reported on in [38] by applying a new form of fitness selection, producing fitter controllers, and by extensive analysis of the behavior and competence of several populations of controllers evolved under different environmental conditions.

Additional results are presented analyzing the course of evolution and the acquisition of behavior over the course of evolution.

The paper is organized as follows: The remainder of the Introduction presents a review of related research, and a survey of common types of fitness selection functions used in evolutionary robotics. Section II presents the physical robot platform used in this research and discuses the video sensors used by the robots. Section III presents the evolutionary neural network architecture and section IV defines the selection criteria used to drive controller evolution. Sections V and VI present the results and testing of evolved game-playing robot controllers evolved under varying conditions.

B. Related work

The field of ER has been reviewed in several publications [20,30,32,43]. Much the research reported on to date has investigated the evolution of controllers for simple tasks such as phototaxis [19,52], or object avoidance [11,26,36].

Locomotion in combination with obstacle avoidance in legged robots has been the subject of several ER studies [10,18,21,23,26]. In [18] Gruau reports on a cellular encoding scheme for evolvable modular neural networks for legged robot control. Filliat *et al*. [10] were able to evolve efficient locomotion and object avoidance abilities in a hexapod robot using networks of threshold neurons and IR sensors to detect the robot's environment. There, controllers were evolved in simulation and transferred to real robots for testing. Jakobi *et al.* [23] described the use of minimal simulation to evolve behaviors in an eight legged robot with sixteen actuator motors. In [26] Kodjabachian *et al.* describe the incremental evolution of walking, object avoidance and chemotaxis in a simulated six-legged insectoid robot. Finally, in [21] Hornby *et al.* describe the evolution of ball chasing using an 18-DOF quadruped robot.

Peg pushing behaviors were evolved in [22,25]. In those works, the task required that two-wheeled robots to push pegs (small cylinders) toward a light source. Earlier in [27] Lee *et al.* investigated a similar box-pushing behavior using Genetic Programming (GP).

Competitive evolution in which the fitness of one individual may affect the fitness evaluation of another individual represents an important element of the research described in this paper. Several examples of competition in the form of co-competitive evolution have been reported in the literature. Cliff and Miller investigated the co-evolution of competing populations in the form of predator-pray behaviors [6,7]. Other similar works have been reported on in [4,12,20,42]. Direct competitive evolution of controllers within a single population is investigated in [37,38] and is further investigated in the research we report on in this article.

Recently, several somewhat more complex tasks than those mentioned above have been reported. It should be noted that many of these tasks are only marginally more complicated than those achieved in the earliest days of ER. The evolution of robot controllers to perform these relatively complex tasks required using complex hand-formulated fitness functions [40,53].

The most difficult tasks addressed in the literature involve some form of sequential action. Nolfi 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 [40]. In [53] Ziemke 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 [14] Floreano *et al.* report on the evolution of a robot behavior in which robots move to a light and then back to a home zone. Another example of evolving controllers for a more complex task is reported in Tuci *et al.* [51]. There, robot controllers evolve to produce life-time learning in order to predict the location of a goal object based on the position of a light source.

Flocking or group movement behaviors have been also been investigated. Ashiru describes a simple robot flocking behavior in [1]. A two-robot coordination task in which two robots evolve to move while maintaining mutual proximity is reported by Quinn in [47]. Baldassarre *et al.* [2] evolved homogeneous controllers for a task in which 4 robots must move together in a small group toward a light or sound source. In [8] the physical aggregation of small robots into a lager structure is investigated and makes use of a relatively complex hand formulated fitness function.

The main value of the largely proof-of-concept ER work done to date is that it has shown that evolvable controller structures (such as neural networks) can be trained to produce functional behaviors in autonomous robots (self regulating sensory/motor close loop systems). 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 to date has shown that it is possible to evolve complex controllers in the general case or for generalized tasks. Further, in almost every case of ER research presented in the literature, a hand formulated task specific fitness selection function is employed that more or less defines the how to achieve the target task.

C. Selection Mechanism in Evolutionary Robotics: Measuring Fitness

Although developing an experimental research platform capable of supporting the evolutionary training of autonomous robot controllers is a non-trivial task, we make the case that fitness selection is now the major hurdle confronting the further development of ER. Many of the initial concerns and criticisms of the field regarding embodiment and transference from simulated to real robots [21] have been addressed. There are sufficient examples of evolutionary robotics development platforms that have successfully demonstrated the generation of working controllers in real robots, and in addition, there have been numerous examples of successful evolution of controllers in simulation with transfer to real robots (refer to the previous review section for further examples.) However, concerns related to fitness evaluation and fitness selection remain largely unresolved. The most complex evolved behaviors to date include no more than three or four coordinated fundamental sub-behaviors [1,14,40,51]. In each of these cases, the fitness functions used to drive the evolutionary process were fairly complex, and relatively selective for an *a priori* known or pre-defined solution.

Here, we review methods for controller fitness evaluation commonly used in evolutionary robotics. The fitness selection function (objective 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 almost always 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 selection function is no longer able to detect fitness differences between individuals in the evolving population.

1) Tailored Fitness Functions: Tailored fitness functions are task specific hand-formulated fitness selection functions that generally include several sub-functions or terms combined in a weighted sum or product. These include terms that measure simple response behaviors, low-level sensor-actuator mapping functions, and any other factors that the human designer may choose in order to improve fitness selection for a particular behavior.

For most complex autonomous robot behaviors, the optimal sensor-actuator mapping that will generate a given behavior is unknown. However, in very simple cases, these relationships might be known, or can be handcrafted by a human designer. For many of the proof-of-concept ER experiments reported in the literature, sensor-actuator mappings are directly selected for, and appear as terms in tailored fitness functions.

Tailored fitness functions are almost always formulated by trial and error and/or based on the human designer's expertise (and often a combination of both).

The great majority of the ER research reported to date has used tailored fitness functions. Often such research has been aimed at evolving controllers to perform extremely simple tasks. These include phototaxis [19,52] and locomotion with object avoidance [11,26,29]. With difficulty, and with sufficient knowledge of the dynamics of a particular task or behavior, tailored fitness functions can be extended to evolve controllers for somewhat more difficult robot

tasks. For example, in [46] the authors describe the evolution of a coordinated movement task involving several robots.

Additional examples of the formulation and implementation of tailored fitness functions are found in [9,23,45,48].

2) Incremental Fitness Functions: One difficulty arising in controller selection when applying evolutionary robotics toward a complex or difficult task is that early in the evolutionary process, controllers may have no detectably ability to perform the task at hand (the "Bootstrap Problem"). We refer to populations of such controllers as "*sub-minimally competent*". Incremental fitness functions overcome the problem of sub-minimally competent initial populations by augmenting the difficulty of the task during evolution. This is a process of explicit training for simple sub-behaviors followed by training for successively more complex behaviors. A main criticism of the use of incremental fitness functions is that they restrict the course of evolution to such a degree that resulting controllers cannot be considered to have evolved truly novel behaviors. They represent the optimization of hand-designed solutions. Even so, several of the most complex evolved robot behaviors reported on in the ER literature were developed using incremental fitness selection functions. In [17] the authors report on the evolution of a prey capture behavior using incremental evolution (i.e. evolution making use of an incremental fitness function). In that work, the researchers compared their incremental fitness selection to pure aggregate success/failure selection and reported that only the incremental approach was able to produce fit controllers. Further examples of the use of incremental fitness functions in ER include [25-27].

One form of incremental evolution involves augmenting the difficulty of the environments in which the robots must operate while using a single aggregate success/failure fitness function. This is referred to as "*environmental-incremental*" evolution. This form of incremental evolution may not constrain the controllers search space to the degree that evolution must converge on a particular predefined solution. Very little work has been done using pure environmental-incremental evolution. In [34], the authors use this type of selection to evolve controllers for a peg collection task similar to the garbage collection task in [53]. That research shows that environmental-incremental evolution using an aggregate success/failure selection function can produce controllers expressing complex behaviors. However, it is not clear to what degree the selection and augmentation of training environments shaped the final evolved controller population. Other examples include [33,45].

3) Aggregate Fitness Selection: Aggregate fitness functions select for high-level success or failure of the robot(s) to complete a given task or behavior. This type of selection reduces injection of human bias into the evolving system by aggregating the evaluation of benefit (or deficit) of all sub-behaviors into a single binary value. This is often called "all-in-one" evaluation. Aggregate fitness selection had been largely dismissed by the ER community because in many instances initial populations of controllers have no detectable level of overall competence (i.e. they are sub-minimally competent). Hence, pure aggregate selection produces no selective pressure at the beginning of evolution and the process cannot get started. Even so, aggregate fitness selection in one form or another appears to be the only method that can be applied to generate complex controllers in the general case without injecting restrictive levels of human or designer bias into the resulting evolved controllers. A rare example of aggregate fitness selection applied to the evolution of a complex task is found in [42].

For truly complex behaviors, tailored fitness selection, and incremental fitness selection result mainly in the optimization of human-designed controller strategies and are not truly examples of the primary evolution or learning of intelligent behavior. At first glance, this appears to present a

rather bleak outlook for the future of ER. And in fact, as pointed out earlier, ER has progressed very little in the past decade.

It is possible however to overcome some of the problems associated with aggregate selection. One such method involves utilizing intra-population competition to present a continually increasing task difficulty to an evolving population of controllers. This, in conjunction with a form of selection that relaxes a minimal initial human bias later in training is explored in this paper and is shown to result in competent controllers, at least for the task we investigate.

4) Competitive and Co-competitive Fitness Selection: Competitive fitness selection utilizes direct competition between members of an evolving population. Robot controllers compete against one another within the same environment so that the behavior of one robot directly affects the fitness evaluation of another. This concept has received a limited but growing amount of attention within areas of evolutionary computation applied toward the automatic generation of intelligent systems. For example, in [5] neural networks were evolved to play computer checkers at the expert level using pure aggregate win/loss selection.

Examples of co-competitive evolution involving populations of predator and prey robots exist in the literature [4,42]. As noted in those works, 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 [4,42] shows this effect only to a modest degree, but results from other areas of evolutionary computing that suggest that given the correct evolutionary conditions, pure aggregate selection combined with intra-population competition can result in the evolution of very competent systems [5,28].

II. The Evolutionary Robotics Physical Research Platform

This research utilizes the EvBots, a recently developed, computationally powerful colony of small mobile robots [16,35].

A. The Robots

Each robot is 5 in. wide by 6.5 in. long by 6 in. high and is constructed on a treaded wheel base and use skid steering differential drive systems. Each robot is equipped with a PC/104 based onboard computer with x86 class processor. A custom RedHat Linux distribution is used as the operating system and is capable of supporting MATLAB in addition to other high-level software packages. The robots are linked to one another and to the Internet via a wireless network access point. Each robot also supports video data acquisition through a front mounted USB video camera. Fully assembled EvBots are shown in Fig. 1 (a), (b). Each robot is fully autonomous and capable of performing all computing control and data management on board.

(a) (b) (c)

Fig. 1. A fully assembled EvBot (a), with colored shell (b), and the real maze environment with several EvBots (c).

A physical reconfigurable maze environment was constructed for the mobile robot colony (Fig. 1 panel (c)). To facilitate vision-based sensing, the maze was surrounded by a blue backdrop and robots were also fitted with colored skirts.

B. The Vision System: Video range-finding emulation sensors

All sensing of the environment by the robots was accomplished via video. The system used here (reported on in [35]) is a vision-base emulation of a scanning planar laser range finder.

The vision system uses fixed geometric elements and color properties within the robot's physical environment to calculate the ranges of colored "*material types*" such as walls, robots, and other objects. The black maze walls are of a constant height so distance can be calculated from a monocular image taken from cameras mounted on the robots at a fixed altitude. Robots are fitted with colored skirts of fixed width.

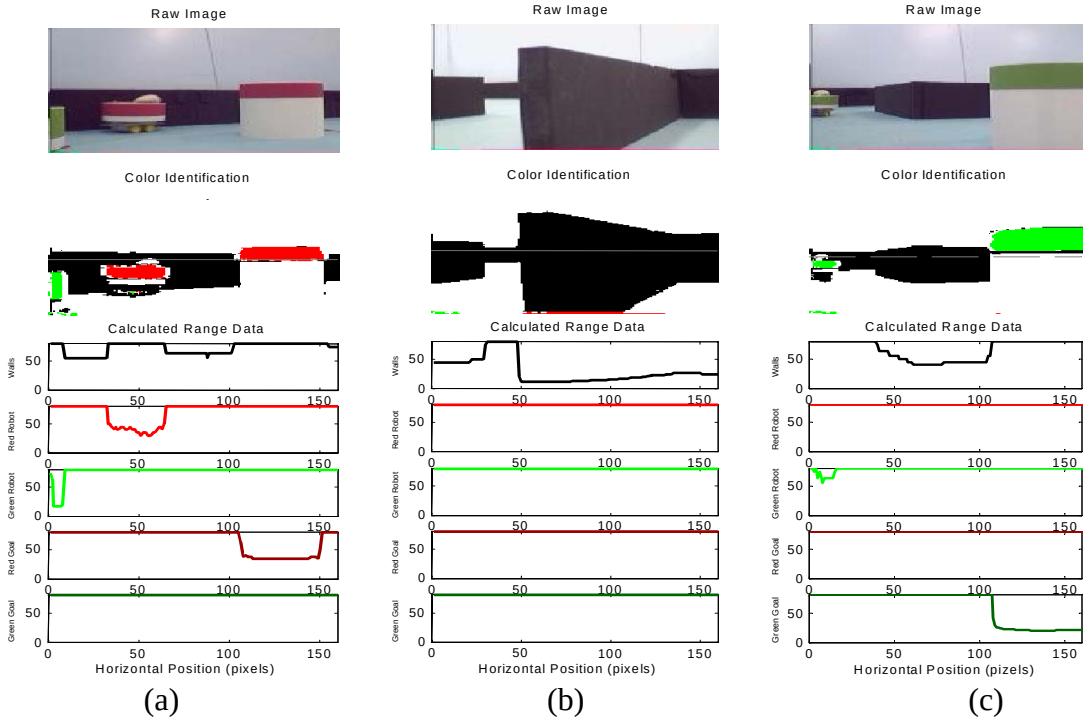


Fig. 2. Examples of image decomposition into vectors of range data to be fed into neural network controller inputs. One vector of length equal to the horizontal resolution in pixels of the image is produced for each 'substance' type in the physical robot environment.

The system successively decomposing video images of fixed resolution. An image is segmented by color (and bisected horizontally to differentiate between robots and goals). It is then converted to a 2D numerical array where the index of each element is its xy-location in the original image and its value is an identifying integer depending on the pixel color.

The vertical sum of pixels, Σp, of each material type is calculated and stored in a set of 5 arrays spanning the horizontal spread of the image. These numerical arrays are then fed element by element through a simple distance formula to produce vectors of ranges d for each material or substance type:

$$d = K \frac{H}{\sum p} \tag{1}$$

Here, H is the physical height of each object type and K is an empirically derived constant.

Fig. 2 shows several example robot-eye-view images and their successive decomposition into range data vectors.

The final output of the vision system is a set of 5 concatenated range data vectors similar to what might be generated by a set of directional range finding sensors that were selective for each type of material. The resolution was set to 1 range value reading per material type per degree, spanning the forward facing field of view of the robot cameras. This generated a total of 150 sensor input values represented by a vector of ordered real numbers. These were fed directly into the robot neural controllers without any bias or indication of origin, beyond consistent ordering. The large number of inputs provide the neural controllers with a high level of potential information about their environment. Other work in the field of ER has generally relied on the equivalent of 5 to 25 photo-detectors, IR sensors, single-input sonar or a combination of these [11,24,31].

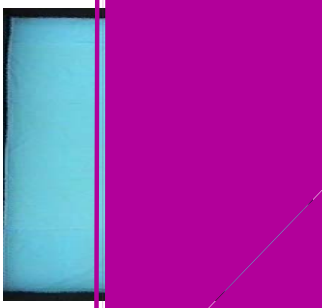
Controller neural networks are only given the resulting numerical data vectors. All associations relating numerical values to physical distances, angles, and substances or objects must be learned by the neural networks during evolution.

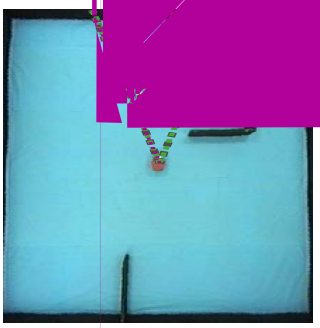
C. Simulated vs. Real Sensors

The real vision system was closely coupled to a simulation environment in which the evolution of the neural controllers was performed [37].

Fig. 3 (a) shows an image of the real maze environment with a graphical representation of real sensor readings superimposed on the image. Here, the sensor data were gathered by the robot in the center of the maze. In panel (b) of Fig. 3, the environment and object configuration is duplicated in simulation. Again, sensor data were taken from the center of the simulated maze and from the same orientation as the real robot in the real maze and superimposed onto the simulated maze graphic. Panels (c) to (f) of Fig. 3 show additional comparative pairs of simulated and real sensor data sets. Over a test set of 10 similar such configurations, the real vision based sensors produced an error of about 12.5 percent when compared to simulated sensor values.

A.L. Nelson, E. Grant, “Using direct
Autonomous Systems, vol. 54, no. 10,





the graphical representation, only nonzero weights are shown (as weighted lines). Neuron location is a function of connectivity. Note that the network shown in Fig. 4 is much smaller than the typical network evolved in this work and is included to illustrate network representation.

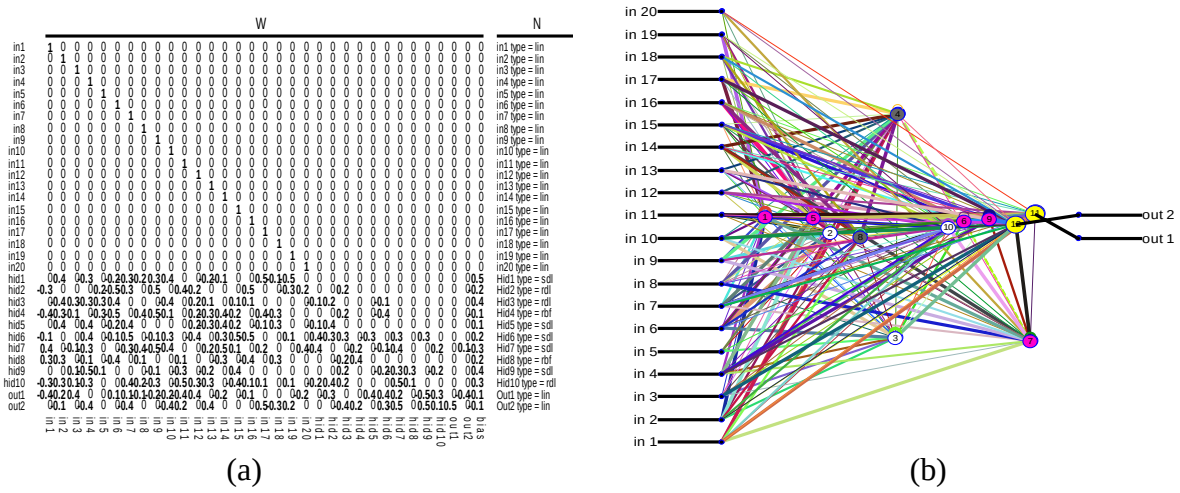


Fig. 4. Example neural network encoding. Panel (a) shows the weight and connectivity matrix **W**, and the neuron type vector **N**. Panel (b) shows the corresponding network graphical representation with inputs on the left and outputs on the right.

This matrix-based network representation is designed to facilitate the evolution of populations of variable-size and arbitrarily connected networks.

Current and past network inputs and activation levels (outputs) are stored in an ordered matrix, **I**. The maximum level of time delay is a scalar integer, δ. Neuron activation functions take the general form:

$$f_n(u) = f_n(\mathbf{w}_n, \mathbf{i}(t, \tau(n))) \tag{2}$$

where $n \in \{1..N\}$, $\mathbf{w}_n$ is the nth row of the weight matrix **W**, $\mathbf{i}(t, \tau(n))$ is the τth row ($1 \le \tau \le \delta$) of the input/activation matrix **I** at time t, and f_n is the activation function type specified in the nth field of **N**. The integer valued time delay, $\tau(n)$ is also defined in the nth field of **N** and is written as a function of n. In most cases, the argument of the neuron activation function, u, takes the form of the sum of the weight inputs (dot product),

$$u = \sum_{m=1}^{N+1} w_m i_m \tag{3}$$

For the radial basis activation functions, u is the Euclidian distance between **w** and **i** in n-space.

Network inputs (from the robot vision system) are calculated as linear neurons while function outputs are selected and read from the matrix **I** after a network updating cycle. The network input-output relationship is:

$$\mathbf{I}(t+1) = Network(\mathbf{I}(t), \mathbf{N}, \mathbf{W}) \tag{4}$$

and

$$\mathbf{o} \subset \mathbf{i}(t+1,1) \tag{5}$$

where **o** is a vector of values from specified output neurons and is a sub-set of ***i***, the first row of the new **I**(t+1). Initially, the network inputs are read into the first elements of the first row vector of **I**(t). **I**(t) is given in expanded form below.

$$\mathbf{I}(t+1) = \begin{bmatrix} [i_1, \ldots i_L, f_{L+1}(\mathbf{w}_{L+1}, \mathbf{i}'(t+1, \tau(L+1)), \ldots f_N(\mathbf{w}_N, \mathbf{i}'(t+1, \tau(N))] \\ \mathbf{i}(t,1) \\ \mathbf{i}(t,2) \\ \vdots \\ \mathbf{i}(t, \delta-1) \end{bmatrix} \tag{6}$$

The functional *Network* of Equation (4) calculates the activations of each neuron specified in **N** in order, placing resulting values in successive elements of **I**.

A fully evolved controller network is shown in Fig. 5. The network uses 150 inputs to accommodate processed video sensor information and produces two drive wheel command outputs that control the robot's differential-steering wheel motors.

IV. FITNESS, SELECTION AND EVOLUTION

This section presents the evolutionary algorithm and fitness selection methods used to train the robot neural network controllers in this research.

Although the issue of fitness function specification is absolutely fundamental to the application of ER to complex problems, it has been largely overlooked or glossed over by researchers up to this point. Two important conflicting factors arise during the process of fitness function specification. These are 1) the need to select for fitness in initial populations that have no measurable ability to complete an overall (complex) task, and 2) the desire to produce fitness measures with a minimum of human bias. If individuals in a population have no detectable level of fitness, then they cannot be ranked in order from fittest to least fit. In such a case, selection is no better than random and artificial evolution cannot proceed (the Bootstrap Problem). The second factor, the need to limit the injection of human bias into evolved solutions, is a requirement (rather than a desire) if ER methods are to be extended to general non-trivial problems in which humans have incomplete domain knowledge. Most ER work to date has neglected the second factor in order to address the first. Indeed, for very simple behaviors, on which much of the early ER work was based, the issue did not arise at all. The reason for this is that for very simple tasks, initial controller populations generally do have some detectable and differentiable fitness with regard to completing the overall task. Unfortunately, this is rarely the case for complex and non-trivial tasks.

In this work we formulate a method of fitness selection to address both of these issues.

A. Fitness Measurement

Fitness for the evolving controllers was based on performance in tournaments of the game *Capture the Flag* (a competitive team game). Each generation consisted of a single tournament involving all of the controllers in the evolving population, followed by selection and propagation of the fittest controllers to the next generation. A bimodal training fitness selection function was used. The selection function has an initial 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 selection metric was applied in a relative intra-population competitive form in which controllers in the evolving population competed directly against one another to complete their task - to win the competitive game *Capture the Flag*.

Fitness $F(p)$ of an individual p in an evolving population $\mathbf{P}$ ($p \in \mathbf{P}$) takes the general form:

$$F(p) = F_{\text{mode\_1}}(p) \oplus F_{\text{mode\_2}}(p) \tag{7}$$

where $F_{\text{mode\_1}}$ is the initial minimal-competence mode and $F_{\text{mode\_2}}$ is the purely success/failure based mode. Here $\oplus$ indicates dependant exclusive-or: if the success/failure based 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 in the case that $F_{\text{mode\_2}}$ is active. $F_{\text{mode\_2}}$ in contrast returns positive values based only on number of games won, if any.

The first mode of the fitness function selects for minimal competence to successfully complete the task (however poorly) in a detectable fraction of the trials, and in a finite amount of time. In essence, the mode selects for the ability to travel a distance *D* through the competition environment. The general form of mode 1 is as follows:

$$F_{\text{mode\_1}} = F_{dist} - s - m \tag{8}$$

where F_{dist} calculates a penalty proportional to the difference between distance *d* travel by the best robot on a team, and the minimal-competence distance *D*:

$$F_{dist} = \begin{Bmatrix} -\alpha * (D-d) & \text{if } d < D \\ 0 & \text{otherwize} \end{Bmatrix} \tag{9}$$

D is defined as half the length of the training environment's greatest dimension and α is a constant of proportionality. In Equation (8), *s* and *m* are penalty constants applied in the case that robots on a team becoming immobilized or stuck (by any means), and, in the case of controllers, producing actuator output commands that exceed the range of the actuators (the wheel motors) respectively. Once the minimal-competence mode is satisfied, it provides no further selective pressure. Note that even the minimal competence mode is nearly devoid of human bias. It says that at least one member of a team of robots should be able to travel through its environment half way without getting stuck. There is no bias encoded into the metric for how this might be accomplished whatsoever, and the metric encodes nothing about the sensors or their relationship to the robot's environment.

The second mode of the fitness function $F_{\text{mode\_2}}$ is classified as purely aggregate because it produces fitness based only on success or failure of the controllers to complete the overall task. Here, competitive games were played, and success or failure was determined by winning or losing games.

For each generation, a tournament of games involving all the individuals in the population was conducted. Each game in the tournament involved two controllers taken from the population of evolving controllers. Each of the competing controllers was instantiated (copied) into all of the robots on a given team. Hence, all the controllers on a particular team had homogeneous controllers. Further, each controller played two games per tournament for a score. (details of tournament and initial conditions, and environmental settings are covered in the Section V. Controllers were awarded fitness points based on the number of wins achieved during each tournament.

The possible combinations of win/lose outcomes of these reciprocal games incurred different levels of fitness and are summarized in Table I.

Table I. Fitness points awarded by the aggregate success/failure mode $F_{\text{mode\_2}}$, for pairs of reciprocal games during a generational tournament.

Game Pair Outcomes	**Fitness Points Awarded**
win-win	**3**

win-draw	**1**
win-lose	**.5**
no win	0 (F_{mode_1} dominates)

Note that in cases where no win occurs F_{mode_1} is used to determine a negative fitness value.

B. Neural Network Controller Selection and Propagation

After each tournament of games (one generation), controller population members p were scored relative to each other using the performance metric $F(p)$ defined in (7), and then propagated to the next generation.

During the propagation phase, controllers from the fittest 50% of the population were copied unchanged to the next generation. This same fittest 50% was then used to produces offspring that replaced the least-fit 50% of the population. An important ramification of this is that in the case that 50% or more of the population received a positive fitness value, then selection will have been based entirely on success/failure information (F_{mode_2}) and the minimal competence mode will have had no bearing, i.e. all individuals not achieving success will have been eliminated.

During evolution, **W** and **N** for each controller neural network in the population were operated on directly. No further genetic encoding or genome was used. The genome **C** is then specified by the two dimensional matrix of real numbers

$$\mathbf{C} = [\mathbf{W} : \mathbf{N}'] \qquad ($$

A. Evolutionary Conditions

Populations of robot controllers were evolved for the ability to play the competitive team game *Capture the Flag*. Games were played by two teams of robots operating in rectangular maze worlds. Robot controllers were monolithic evolvable neural networks of arbitrary connectivity, as discussed previously.

Controller evolution was performed under several environmental and evolutionary conditions. All-in-one, verses environmental-incremental evolution was investigated. Additionally, the effects of applying different methods for choosing which controllers competed against one another during each generation/tournament were investigated.

For comparison, four separate populations were evolved using different evolutionary conditions. These conditions are discussed below.

All-in-one vs. Incremental Evolution: All-in-one evolution was compared to environmental-incremental evolution.

In the incremental evolution case, robot controller populations were evolved in increasingly complex worlds over the course of evolution (the *environmental-incremental* case). Over the course of evolution, as populations attained a level of competence in one world, they were graduated to progressively more challenging training worlds. The sequence of training worlds used is shown in Fig. 6 below.

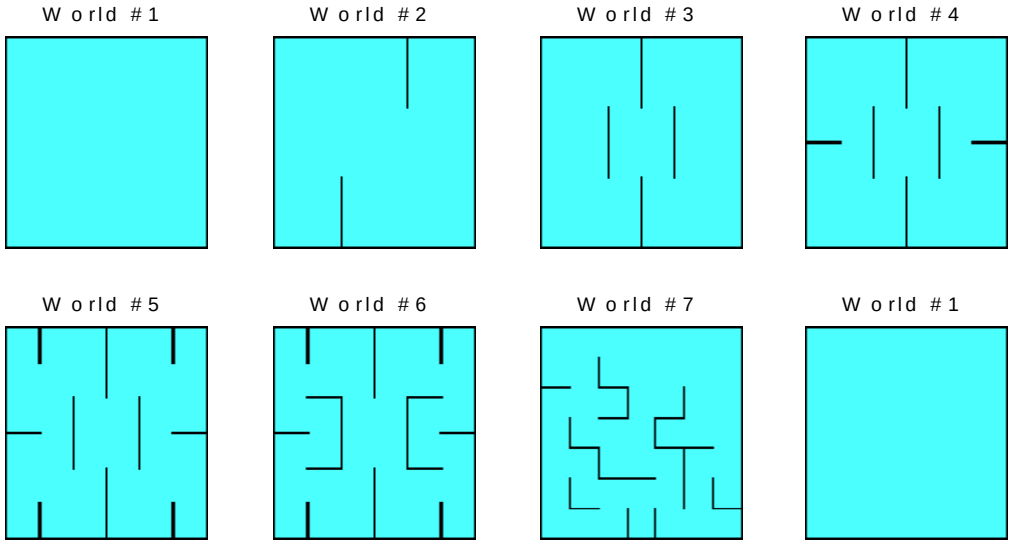


Fig. 6. Training worlds used over the course of evolution.

Populations were also evolved from start to finish in a single very challenging world (world #7 from Fig. 6). This will be referred to as the *all-in-one* case.

Controller Opponent Selection: During evolution, all the games within a particular generation/tournament had identical initialization conditions (starting positions of robots, goals (flags), and so on). At the beginning of each generation/tournament though, a new game initialization was randomly generated. During each tournament/generation every controller in the population played two games for a score against another member of the population (the *opponent controller*).

Two different methods were used to select opponent controllers. For the first method of opponent selection, one specific controller was picked at random from the current population and served as the competitor (*opponent controller*) for every game played during that particular generation/tournament (the *constant opponent* case) In the second method, every new game used a new randomly selected opponent controller from within the current population (the *random opponent* case).

Pairing these factors (*all-in-one* vs. *incremental evolution*, and *constant* vs. *random opponent selection*), we have four separate evolutionary conditions: 1) Evolution in a single difficult world with a single constant opponent being used for every game in a tournament/generation; 2) Evolution in a single difficult world but with a new opponent being randomly selected for each game; 3) Evolution in worlds of incremental difficulty with a single constant opponent being used through a tournament; and 4) Evolution in incremental worlds with a new opponent selected for each game. Populations were evolved under each of these conditions (summarized in Table II). All conditions except those listed in Table II were kept constant.

Table II. Summary of the evolution conditions used to evolve four populations of autonomous robot controllers.

Population Name	**Evolution Environments**	**Opponent selection**
Population 1	Single world	Constant opponent
Population 2	Single world	Random opponent
Population 3	Incremental worlds	Constant opponent
Population 4	Incremental worlds	Random opponent

All populations were kept at a constant size of 40 networks. The four populations were initialized with the same random seed set of neural controllers. Network weight mutation rates were set at 25%, and weight mutation magnitudes were selected from the linear distribution (-1, 1). During mutation networks had a 70% chance of adding or removing a neuron or connection. Networks had a fixed number inputs to accommodate the video sensors (150 for this work) and 2 outputs (for the robot differential steering wheel actuators).

A simple elitist 50% selection and replacement propagation mechanism was used. However, it should be noted that the competitive fitness selection method introduces a high degree of non-determinism into the processes and the evolutionary process maintains a probabilistic element because of this (i.e. there is a chance that the best controller might not be selected, if it competes in particularly unfavorable games). Table II lists settings and parameters.

Table II. Parameter settings common to all of the evolved populations.

Parameter	Setting
Population Size	40
Sensor inputs neurons	150
Initial Internal neurons	60
Chance of adding or removing a single neuron (during network mutation)	70%
Weight initialization range	[-1 1], linear distribution
Weight mutation magnitude range	[-1 1], linear distribution
Weight mutation rate	25%
Initial feedforward connectivity level	60%
Initial feedback connectivity level	20%
Chance of adding or removing a single connection (during network mutation)	70%
Elitism level (per generation)	Single best from previous generation
Population replacement rate (per generation)	1/2 + 1/40
Generations (per evolution)	650

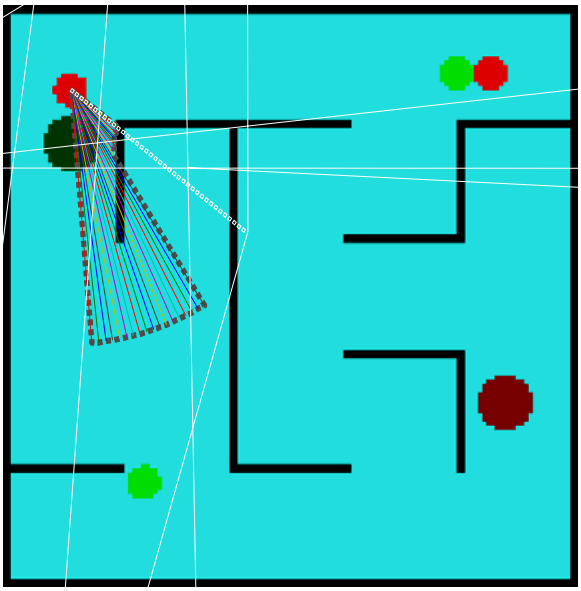
The computation time for the evolutionary trainings performed, were relatively long - between 3 and 4 weeks of computer runtime per population (using a 2.7 GHz. Pentium III desktop computer). Hence, it was impractical to perform more than one evolution per environmental condition.

B. Demonstration and Discussion of Evolved Controller Capabilities

All four of the evolved populations produced competent controllers after 450 generations. Here, we focus qualitative analysis only on population 4. In section VI all of the populations will be compared one to another and it will shown that population 4 is measurably better than the others, although not to a dramatic degree.

Fig. 7 shows two example games of robotic *Capture the Flag* played between controllers taken from the 450th generation of Population 4. These games were conducted in simulated maze environments novel to the evolved controllers (i.e. not seen during evolutionary training). In each panel of Fig. 7, the smaller dots with the fan-like graphics are the robots. The fan-like graphics display sensor data and are not physical objects. The paths taken by the robots during the games are indicated by the irregular line-curves. There are two robots on each team to make a total of four robots in each game. The larger dots represent the stationary goal objects (flags). The heavy black line segments represent walls in the environments. In panel (a) the red team robots were controlled by copies of the best network in the population and the green team robots by copies of the second best network. In this case, the game was won by the red team (paths indicated by the dark lines). The red robot that won the game for its team has made its way to the green goal object in the upper left corner of the maze.

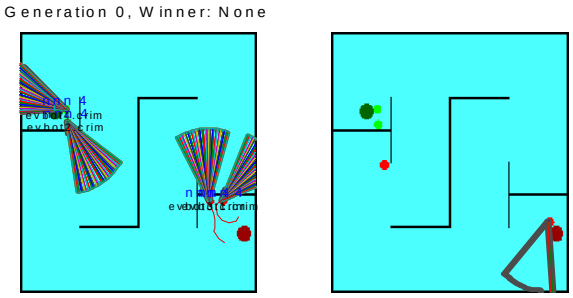
A.L. Nelson, E. Grant, "Using direct competition to select for competent controllers in evolutionary robotics," Robotics and Autonomous Systems, vol. 54, no. 10, pp. 840-857, Oct. 2006.



C. Acquisition of Competence over the Course of Evolution

A set of experiments was performed to investigate acquisition of behaviors by the controllers over the course of evolution. A set of simulated games was played using controllers taken from different generations spanning the course of evolution of population 4. Fig. 8 shows a series of eight such games, each involving controllers taken from advancing generations of the population.

In the first game (Fig. 8 panel (a)), the original progenitor controller from the initial un-evolved neural network population was used. In the remaining games, the best controller from each generation being tested was selected and copied into the competing robots. In the second game, the best controller from generation 50 was used. In the subsequent panels (c) to (h), 100 generations progressively separated the "ages" of the populations.



(a)

Table IV. Qualitative acquisition of behaviors over the course of evolution of population 4. The solid dots indicate that a behavior is observed in that generation. The open dots indicate that the behavior has been superseded by another.

Generation	Rudimentary wall avoidance	Effective simple navigation: (always turn left)	Avoid own goal	Qualitatively complex navigation: (left and right turns)	Left-hand mouse search strategy
0					
50	●				
150	●	●			
250	○	●	●		
350	○	○	●	●	
450	○	○	●	●	●
550	○	○	●	●	●
650	○	○	●	●	●

Table IV summarizes the discussion of acquisition of behaviors over the course of evolution of population 4. The identification of a particular behavior here represents a qualitative human assessment based on observation of robots during game sequences. In the later generations, exact behavior is very difficult to predict, and behaviors observed from the distal (exterior to the robot-controller system) are not necessarily reducible to desecrate behaviors at the proximal level (from the point of view of the robot controller). Actual behaviors at the proximal level are likely to be inextricable co-coupled with one another, and with sensor inputs [41].

VI. Measuring the Performance of Evolved Controllers

A. The Application of Metrics for Post Evolution Evaluation of Evolved Controllers

The experiments presented in this section focus on defining and applying absolute metrics for evaluating the performance of evolved robot controllers. In [6] it was noted that co-evolutionary processes do not evolve within the context of a pre-determined or fixed fitness landscape. The fitness of each individual in a population will have been affected by the fitnesses of the other individuals in that population, thus producing a changing fitness landscape with varying, non-absolute fitness values. Because of this, absolute fitness values cannot be used to monitor improvement past the initial phases of evolution.

B. A Metric for Post-evolution Evaluation of Controller Performance

To measure absolute performance, evolved controllers were compared to a controller of known abilities. A knowledge-based controller was designed by hand to play the robotic version

of *Capture the Flag*. Table V summarizes the knowledge-base controller's behaviors in order of precedence.

Table V. Behaviors expressed by the hand-coded knowledge-based controller. Behaviors are given in order of precedence.

Precedence	Behavior
1	If opponent goal is detected, robot actively homes in on goal
2	Turn away from own goal
3	Avoids teammate robots by passing on the right
4	Block close opponent robots
5	Avoid distant opponent robots
6	Extract from corners and close walls
7	Avoid walls and follow corridors if no other objects are detected
8	Move forward in open space
9	Detect immobilization on out-of-view objects and respond with rotations of random degree

We used the knowledge-based controller to evaluate the acquired game-playing abilities of the four populations that were evolved under different environmental conditions (listed in Tab. II of Section V). The "best" performing controller from each population was pitted against the knowledge-based controller in a tournaments of 240 games. Each tournament used 120 random robot-and-goal position initializations based on the first 120 random numbers generated by MATLAB 5.3 using the initial seed 1. To eliminate possible advantage to one or the other controller (robot team), each initial game position was used for a set of two reciprocal games. In the first game, the neural network controllers controlled team 1 (red), while the knowledge-based controllers controlled the team 2 (green). Visa versa, in the second game, the knowledge-based controllers controlled the team 1 robots, while the team 2 robots were controlled by the neural network controllers.

These tournaments were carried out in three different maze world configurations (shown in Fig. 9). Evolved controllers were tested in multiple environments so that a more accurate overall evaluation of their capabilities could be made.

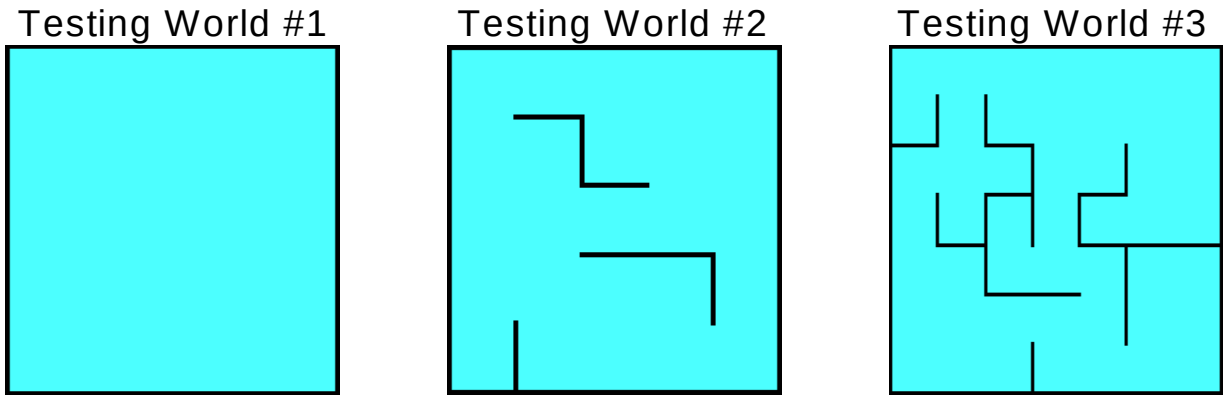


Fig. 9. Three testing maze world configurations.

The knowledge-base controller was not involved in any way with the selection or evolution of the evolved controller populations and its behavior had no effect on evolving controller strategies.

Selection of the “best” individual in a population was done by averaging the net number of wins achieved by each contro

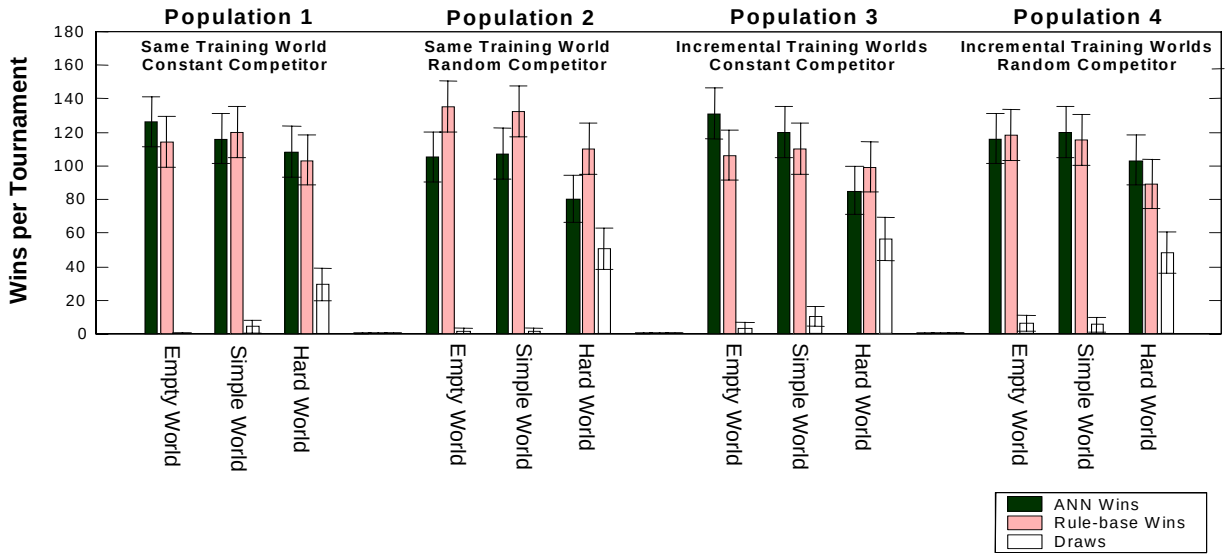


Fig. 10. Four populations evolved under different conditions were evaluated in competition with the knowledge-based controller. The best individual from each of the populations played 240 games against the knowledge-based controller in each of the three testing worlds. Each sub-triplet of bars gives the results from a single tournament in a single world. The dark bars indicate the number of evolved controller wins in a given tournament, the shaded bars show the number of wins achieved by the knowledge-base (rule base) and the white bars indicate games that ran over the time limit (draws).

The best controllers from all four of the evolved populations were able to win a substantial number of games in all of the competitions. However, only controllers populations 1 and 4 were able to win more games than the knowledge-based controller.

At first glance, the data shown in Fig. 10 indicate that the environmental-incremental evolution produces somewhat superior results. However, population 1, which was evolved entirely in world #7 of Fig. 5.1 (the most difficult training world) was able the win a significant number of games in each of the test worlds and also out-competed the knowledge-based controller, although not to a statistically significant degree. Although the best results were achieved in population 4 using environmental-incremental evolution, these results show that environmental-incremental evolution, is not absolutely required at least in the case of the behaviors evolved in this work.

The most salient result of the data shown in Fig. 10 is that by the 450th generation, populations in three out of the four evolutionary conditions were able to produce behavior that was competitive with the knowledge-base in all of the testing worlds.

It was considered that random competitor selection during training tournaments would add excessive noise to the selection process and result in poorer or slower absolute fitness evolution. This is in fact seen in the absolute fitness's of population 2, as shown in the second super-group of bars 9 in Fig. 10. There the evolved neural network controller was beaten by a statistically significant number of games by the knowledge-based controller in the first and third testing worlds. However, the same effect was not observed when random opponent selection was applied to the environmental-incremental case of population 4. That evolution appeared to produce the fittest "best" controllers of all of the evolution conditions.

The metrics for absolute controller fitness of Fig. 10 do give a numerical relationship between these particular four "best" evolved controllers with respect to the knowledge-based controller and to one another. However, those data only weakly indicate differences due to evolutionary conditions because full evolutions starting from random initial controllers were performed once for each of the four conditions.

What can be said is that incremental and non-incremental conditions both produce competent controllers. These results disagree with those reported in [6]. However, the selection methods in [6] were not comparable to those used in this research.

D. Measuring Absolute Fitness Over the Course of Evolution

Competition of evolved controllers against the known knowledge-based controller was also used to measure the developing absolute fitness of evolving populations over the course of evolution. Again, here we focus just on one particular population (population 4). Fig. 11 shows the progression of acquisition of game playing ability over the course of evolution of population 4. The figure shows competitions performed with the best current individual taken every 100 generations starting with generation 50 and ending with generation 650. Each generation was tested in the three test worlds (see Fig. 9), and in each case 240 games were played against the knowledge-based controller.

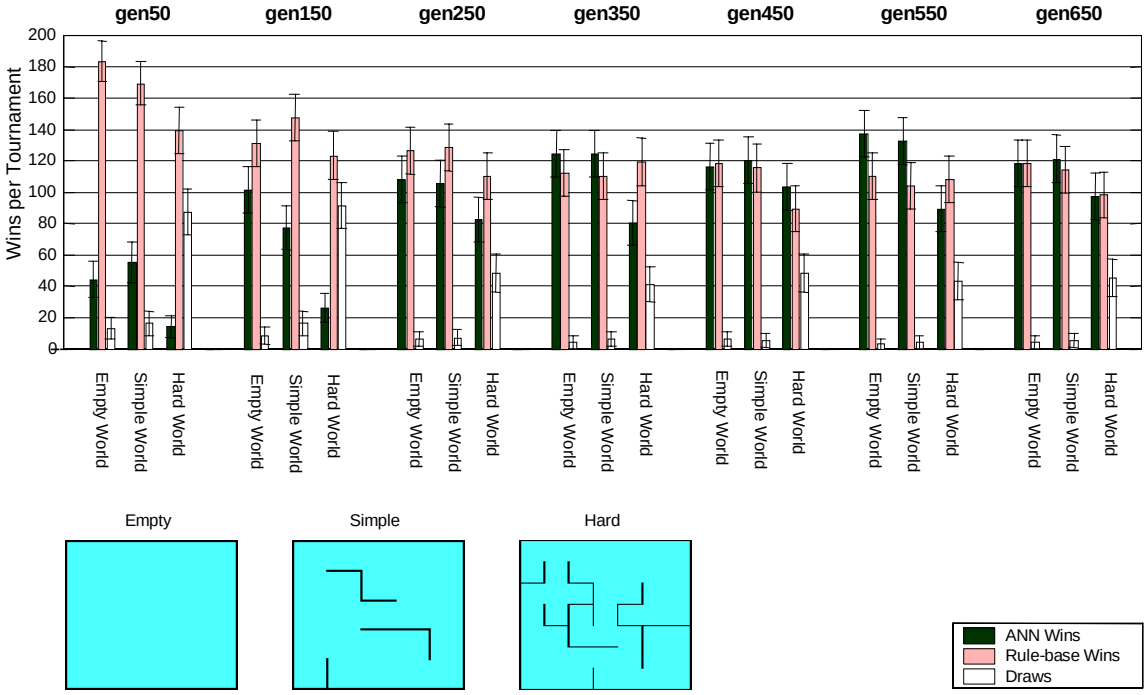
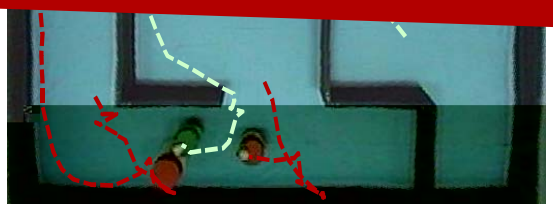


Fig. 11. Performance of population 4 in competition against the knowledge-based controller at successive generation points during the course of evolution. Each best controller played one tournament in each of the three testing worlds. The testing worlds are repeated for reference from Fig. 9 in the thumbnail insets in the lower left of the figure.

The sequence of grouped tournament results shows a steady increase in game playing ability. This is especially prominent in the most difficult test world. At generation 50, the best individual in the evolving population was only able to win 14 games out of 250 while the Knowledge-based controller won 139 games. At generation 450, the evolving controllers were able to play competitively with the Knowledge-based controller, wining 103 games to the Knowledge-based's 89. Note that competitive play in the empty world is achieved much earlier by the 150



It is important to demonstrate that differences between simulation and reality are small enough so that behaviors evolved in simulation are expressed in the physical world. In order to achieve this, it is important to develop ER simulation test-beds in conjunction with physical robot systems. Although it may be possible to develop a simulation environment without a particular real robot system in mind, it is much more difficult than early researchers may have believed. We believe the difficulties stem mainly from conceptual problems rather than technological ones. The design of unrealistic sensors, the incidental use of un-obtainable environmental information, and other similar issues can lead to the development of untransferable controllers [17,44]. For these reasons, we believe that the feasibility of transference is best demonstrated by example. This will likely not be the case in the future, in light of the rapid development of virtual reality, gaming physics engines and the rise of realistic computer animation for entertainment. In the early years of ER research, it was speculated that transference problems could only be overcome by evolving controllers using only real robots [3]. However, and perhaps surprisingly, this did not turn out to be the case. It is in fact well within the current state of the art of computing, simulation, an autonomous robot technology to build finely couple simulation and hardware platforms. The real problems lie in the most fundamental elements of the process of automatic construction of intelligence: selection of fitness in the non-trivial case.

In the research reported on in this paper, the behaviors of the knowledge-based controller were known, and could be observed in the real world and in simulation. Based on qualitative comparisons, the behaviors of the hand-coded controller were observed to be expressed similarly in simulation and reality. As is the case with the evolved controllers, the hand-coded controller could be run on stimulated robot agents, and the physical robots without the need for any modifications. Hence, its qualitative predictability in both simulated and real environments does support the assertion that behaviors do transfer from simulation to reality in this robot research platform.

That said; transference was not perfect. Because of the iterative looping nature of the controller structure used, both the evolved and hand-code controllers re-compensate for slightly mis-calibrated sensors and actuators. These slight mis-calibrations were observed to lead to transference problems in one situation. If a robot controller produces a precisely evolved arch command close to a right-angled corner or wall edge, the robot can move in one time step into a position where the robot is stuck, but in which the robot camera does not see the object the robot is stuck on. It is important to note that the condition of colliding with objects that might be out of view of the camera is modeled in the simulated world, and in general, the controllers evolved to avoid this.

VII. CONCLUSIONS

This paper has presented the evolution of autonomous robot controllers for playing a competitive team game. The work focused on the development and application of fitness selection methods that minimize the level of *a priori* task knowledge needed to generate competent controllers. In particular we investigated aggregate success/failure selection in combination with intra-population competition to drive the evolution of complex behaviors in autonomous systems.

Robot neural network-based controllers were evolved under several environmental conditions. Evolved networks contained feedforward and feedback connections, 150 or more processed

video sensor inputs and on the order of 100 internal neurons of mixed type. Connection weights, connectivity and neuron structure were all evolved during training.

Evolved controllers were evaluated qualitatively over the course of evolution in several experimental studies. Fully evolved controllers were evaluated quantitatively by comparing their performance in extensive competitions to that of a knowledge-based controller of well defined abilities,

Further work will include investigations into a broader range of environmental and algorithmic conditions and for robots using a greater range of actuators and sensors. In addition, it would be desirable to investigate the degree to which relative competitive aggregate selection can be applied without any bootstrap mode toward environmental-incremental evolution of controllers for complex tasks.

REFERENCES

[1] 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.

[2] G. Baldassarre, S. Nolfi, D. Parisi, Evolving mobile robots able to display collective behaviours, in: C. K. Hemelrijk, E. Bonabeau, eds., Proceedings of the International Workshop on Self-Organisation and Evolution of Social Behaviour, Monte Verità, Ascona, Switzerland, 2002, pp. 11-22.

[3] R.A. Brooks, Elephants don't play chess, Robotics and Autonomous Systems 6 (1990) 3-15.

[4] G. Buason, T. Ziemke, Competitive Co-evolution of predator and prey sensory-motor systems, in: Proceedings of the second European workshop on Evolutionary Robotics: Applications of Evolutionary Computing, G. Raidls et al. eds., Springer-Verlag, Berlin, Heidelberg, 2003, pp. 605-615.

[5] 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.

[6] D. Cliff and 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., Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life (ECAL'95). Lecture Notes in Artificial Intelligence 929, Springer-Verlag, 1995, pp. 200-218,

[7] D. Cliff and G. F. Miller, Co-evolution of pursuit and evasion II: simulation methods and results, in: P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, W. Wilson eds., From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB'96). MIT Press/Bradford Books, pp. 506-515, 1996.

[8] 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.

[9] J.A. Driscoll, R.A. Peters, II, A development environment for evolutionary robotics, in: Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics, Vol. 5, 2000, pp. 3841-3845.

[10] D. Filliat, J. Kodjabachian, J.A. Meyer, Incremental evolution of neural controllers for navigation in a 6 1egged robot, in: M. Sugisaka, H. Tanaka, eds., Proceedings of the Fourth International Symposium on Artificial Life and Robotics, Oita University Press, 1999.

[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] D. Floreano, S. Nolfi, F. Mondada, Competitive co-evolutionary robotics: from theory to practice, in: R. Pfeifer, ed., Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB'98), From Animals to Animats 5, Cambridge, MA, MIT Press, 1998.

[13] D. Floreano, J. Urzelai, Evolutionary robots: the next generation, in: T. Gomi, ed., Proceedings of the 7th International Symposium on Evolutionary Robotics (ER'00): From Intelligent Robots to Artificial Life, AAI Books, 2000, pp. 231-266.

[14] D. Floreano, J. Urzelai, Evolutionary robots with on-line self-organization and behavioral fitness, Neural Networks 13 (4-5) (2000) 431-443.

[15] L. Fogel A. Owens, M. Walsh, Artificial Intelligence Through Simulated Evolution, John Wiley & Sons, Inc, New York, 1966.

[16] J.Galeotti, S. Rhody, A. 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'02), San Diego Ca, 2002, pp. 86-91.

[17] F. Gomez, R. Miikkulainen, Incremental Evolution of Complex General Behavior, Adaptive Behavior 5 (1997) 317-342.

[18] F. Gruau, "Automatic definition of modular neural networks, Adaptive Behavior 2 (1995) 151-183.

[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, Proceedings of 3rd International Conference on Simulation of Adaptive Behavior, SAB94, MIT Press/Bradford Books, Boston, MA, 1994, pp. 392-401.

[20] 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.

[21] 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., Proceedings of the Third International Conference on Evolvable Systems: from Biology to Hardware; (ICES'00), Lecture Notes in Computer Science 1801, Springer, 2000, pp. 80-89.

[22] 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.

[23] 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.

[24] N. Jakobi, P. Husbands, I. Harvey, Noise and the reality gap: The use of simulation in evolutionary robotics, in: F. Moran, A. Moreno, J. Merelo, P. Chacon, eds., Advances in Artificial Life: Proceedings of the 3rd European Conference on Artificial Life, Lecture Notes in Artificial Intelligence, Vol. 929, Springer-Verlag, 1995, pp. 704-720.

[25] 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 (ICRA'01), Vol. 4, Maui, HI, 2001, pp. 2384-2391.

[26] 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 9 (5) (1998) 796-812.

[27] 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.

[28] A. Lubberts, R. Miikkulainen, Co-evolving a go-playing neural network, in: Coevolution: Turning Algorithms upon Themselves, Birds-of-a-Feather Workshop: Proceedings of the Genetic and Evolutionary Computation Conference Workshop, (GECCO'01), San Francisco, CA, 2001, pp. 14-19.

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

[30] M. Mataric, D. Cliff, Challenges in evolving controllers for physical robots, Robotics and Autonomous Systems 19 (1) (1996) 67-83.

[31] D. Marocco, D. Floreano, Active vision and feature selection in evolutionary behavioral systems." in: J. Hallam, D. Floreano, G. Hayes, J. Meyer, J. eds., Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior: from Animals to Animats 7, MIT Press, Cambridge, MA, 2002, pp. 247-255.

[32] 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.

[33] O. Miglino, D. Denaro, G. Tascini, D. Parisi, Detour behavior in evolving robots: are internal representations necessary?, in: Proceedings of First European Workshop on Evolutionary Robotics, Springer-Verlag, 1998, pp. 59-70.

[34] H. Nakamura, A. Ishiguro, Y. Uchilkawa, Evolutionary construction of behavior arbitration mechanisms based on dynamically-rearranging neural networks, in: Proceedings of the 2000 IEEE Congress on Evolutionary Computation, Vol. 1, 2000, pp. 158-165.

[35] 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 (IROS,03), Las Vegas NV, Vol. 3, 2003, pp. 2273-2278.

[36] A.L. Nelson, E. Grant, J.M. Galeotti, S. Rhody, Maze exploration behaviors using an integrated evolutionary robotics environment, Robotics and Autonomous Systems 46 (3) (2004) 159-173.

[37] A.L. Nelson, E. Grant, T.C. Henderson, Competitive relative performance evaluation of neural controllers for competitive game playing with teams of real mobile robots, in: Proceedings of the 2002 PerMIS Workshop: Measuring the Performance and Intelligence of Systems, Gaithersburg, MD, 2002, pp. 43-50.

[38] A.L. Nelson, E. Grant, T.C. Henderson, Evolution of neural controllers for competitive game playing with teams of mobile robots, Journal of Robotics and Autonomous Systems 46 (3) (2004) 135-150.

[39] A. L. Nelson, E. Grant, G. Lee, Developing evolutionary neural controllers for teams of mobile robots playing a complex game, IEEE International Conference on Information Reuse and Integration, (IRI’03), Las Vegas NV, 2003, pp. 212-218.

[40] S. Nolfi, Evolving non-trivial behaviors on real robots, Robotics and Autonomous Systems, 22 (3-4) (1997) 187-198.

[41] S. Nolfi, Evolutionary robotics: exploiting the full power of self-organization, Connection Science 10 (1998) 167-183.

[42] S. Nolfi, D. Floreano, Co-evolving predator and prey robots: do 'arms races' arise in artificial evolution?” Artificial Life 4 (4) (1998) 311-335.

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

[44] S. Nolfi, J.L. Elman, D. Parisi, Learning and evolution in neural networks, Adaptive Behavior 3 (1) (1994) 5-28.

[45] F. Pasemann, U. Steinmetz, M. Hülse, B. Lara, Evolving brain structures for robot control, in: Proceedings of the 6th International Work-Conference on Artificial and Natural Neural Networks: Bio-inspired Applications of Connectionism-Part II, (IWANN'01), Granada, Spain, Lecture Notes In Computer Science, Vol. 2085, Springer-Verlag, 2001, pp. 410-417.

[46] M. Quinn, Evolving communication without dedicated communication channels, in: J. Kelemen, P. Sosik, eds., Advances in Artificial Life: Proceedings of the Sixth European Conference on Artificial Life (ECAL’01), Prague, Czech Republic, Springer, 2001, pp. 357-366.

[47] 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.

[48] M. Quinn, L. Smith, G. Mayley, P. Husbands, Evolving team behaviour for real robots, in: Proceedings of the EPSRC/BBSRC International Workshop on Biologically-Inspired Robotics: The Legacy of W. Grey Walter (WGW '02), HP Bristol Labs, U.K., 2002.

[49] L. Rade, B. Westergren, Mathematics Handbook for Scientists and Engineers, Birkhauser, Sweden, 1995, pp. 471-476.

[50] F. Southley, F. Karray, Approaching evolutionary robotics through population-based incremental learning, in: Proceedings of the 1999 IEEE Conference on Systems, Man, and Cybernetics, Vol. 2, 1999, pp. 710-715.

[51] 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.

[52] 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.

[53] T. Ziemke, Remembering how to behave: Recurrent neural networks for adaptive robot behavior, in: L.R. Medsker, L.C. Jain, eds., Recurrent Neural Networks: Design and Applications, Boca Raton, CRC Press, 1999.