

# DEVELOPING A DRIBBLE-AND-SCORE BEHAVIOUR FOR ROBOT SOCCER USING NEURO EVOLUTION<sup>1</sup>

*Vic Ciesielski<sup>2</sup> and Shih Yoon Lai<sup>2</sup>*

<sup>2</sup>Department of Computer Science  
RMIT  
GPO Box 2476V  
Melbourne 3001  
vc@cs.rmit.edu.au  
Tel: (03) 9925-2926, Fax: (03) 9662-1617

## ABSTRACT

We have experimented with a neuro evolutionary algorithm that was successfully used in simulated ice hockey (Blair & Sklar 1998) to evolve a player who can execute a 'dribble the ball to the goal and score' behaviour in the more complex environment of robot soccer. We used both goal-only and composite fitness functions. Due to the very high computational cost we worked with a number of restricted and simplified starting positions. The evolved players developed rudimentary skills, however it appears that considerably more computation time is required to get competent players. Also, it appears that a goal-only fitness is more likely to lead to success than a composite fitness function.

## 1 Introduction

RoboCup is an international competition in which teams of robots compete against each other in a soccer game (Kitano, Asada, Kuniyoshi, Noda & Osawa 1997, Noda, Matsubara, Hiraki & Frank 1998). RoboCup is designed to be an environment for the development of co-operative agents operating in a changing, uncertain domain. There are a number of hardware leagues and a simulator league. Our interest is in the simulator league. In this league the games are controlled by a simulator which sends sensory information to the players and accepts and executes commands from the players. Developing a team for the competition is extremely challenging. In addition to dealing with issues of a rapidly changing world, complex physics and autonomous agents, players must also cope with the fact that the information supplied by the simulator is limited and noisy and the simulator does not carry out commands as specified, but adds a random element to each kick, dash and turn command. Our general goal is to build a team using machine learning and evolutionary methods. We expect that a very useful team member will be an attacker who can find the ball, dribble towards the goal and score. This paper describes our initial efforts towards the development of such a player. The behaviour we would like to learn is:

Given only one player on the field at some random starting position and the ball at a random position, but where the player can see it, the player should go to the ball, dribble it towards the goal and score a goal.

A similar goal scoring behaviour was successfully evolved in the Shock ice-hockey simulation (Blair & Sklar 1998, Blair & Sklar 1999). We would like to determine whether the same approach could be successfully used in the more complex robocup environment. The work in Shock was successful in that the evolutionary process successfully generated a trained neural network that could score goals. However, a massive number of fitness evaluations was needed, 200,000-500,000. This number of evaluations is not feasible for the robocup environment, so we have investigated strategies to get convergence in many fewer evaluations. These strategies involve starting with a number of easy situations where the ball and player always start in fixed positions and the goal is in view and working up to harder ones where the player and the ball are initially in random locations and the player may not be able to see the ball or the goal.

A major consideration in using an evolutionary algorithm is the fitness function. Previous work has used goal-only and composite fitness functions. In a goal-only fitness function the number of goals scored in a fixed time period or the time taken to score a goal is used as the fitness measure (Blair & Sklar 1998, Luke 1998). In a

<sup>1</sup>In Peter Whigham et al. (Editors) *Proceedings of The 5th Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, ISBN 0 7317 0503 3, Pages 70-78, University of Otago, 2001

composite fitness function important features such as being near the ball, kicking the ball, being near the goal and kicking a goal are combined into one overall measure of fitness (Andre & Teller 1999).

	Advantage	Disadvantage
Goal-only	Only the important behaviour of goal scoring is rewarded.	It can be a long time before a goal is scored and an individual achieves a <u>non zero fitness</u> .
Composite	Promising individuals can be rewarded early	Finding the right combination and weighting of factors is very difficult.

## 1.1 Aims

The overall aim of this work is to determine whether a player can be trained to learn a dribble-and-score behaviour which involves running to the ball, dribbling towards the goal and kicking a goal. In particular we will investigate whether a dribble-and-score behaviour can be learned in the following situations:

1. Ball in center of field, player close to the ball and facing the goal, composite fitness function (figure 3a).
2. Ball close to the goal, player in three different positions near the ball, player can see the goal, composite fitness (figure 4a).
3. Ball in a random position near the goal, player near the ball but may or may not be facing ball/goal, composite fitness (figure 5a).
4. Ball in a random position near the goal, player near the ball but may or may not be facing ball/goal, goal-only fitness (figure 5a).

## 2 The Robocup Simulation Environment

In the simulator league the soccer server is responsible for providing the artificial world in which games are played. Each player in the simulation is its own process, with communication between players limited to messages which must pass through the soccer server via unix sockets. A full description of the operation of the soccer server can be found in (Noda 1998).

Messages, represented as strings, allow communication between players and the server. Messages sent by players inform the server of actions they wish to execute. Messages from the server inform players of the position of the ball, positions of other players, lines, flags and goals which a player can see on the field. A list of commands used in the evolution is shown in table 1. The full set of commands can be found in (Noda 1998).

Command	Description
<code>(turn <i>Dir</i>)</code>	Changes the direction a player is facing by <i>Dir</i> degrees.
<code>(dash <i>Pow</i>)</code>	Increases the momentum of the player with the power <i>Pow</i> , causing it to run in the current direction.
<code>(kick <i>Pow Dir</i>)</code>	Kicks the ball with a power of <i>Pow</i> and in the direction <i>Dir</i> . Players may only kick the ball if they are close enough to it.

Table 1: Basic RoboCup player control commands

After a game has begun, the server sends updated percept information to each player. This occurs once every five-hundred milliseconds, but players are expected to send actions to the server once every one-hundred milliseconds. Because of the length of time between percept updates, players are often acting up out of date percept information.

Players receive sensory information from the server via a percept vector. Full details of the percept vectors can be found in (Noda 1998). The following example illustrates the basic mechanism. A player who can see the ball 5 units away directly in front of itself, a teammate (number 11) off to its left and an opponent, too far away to its right for its number to be visible, would receive the percept vector:

```
((ball) 5 0 0 0) ((player MyTeam 11) 10 -30 0 0) ((player TheirTeam) 10 0 5))
```

One factor that makes the robocup domain an interesting and difficult one is the randomness added by the simulator. Each parameter value for kick, dash and turn has a random value added to it before being executed. Thus the player cannot rely on an action being carried out exactly as specified and no two games are the same even if they start from identical initial conditions.

### 3 Neural Network Architecture

We have chosen a minimal architecture for the neural network. This is shown in figure 1. The outputs of the network are determined by the possible actions that a player can take. However there are many possibilities for the input nodes as there is vast amount of sensory input available to the player. We have selected what we think is the minimal amount of sensory information needed to generate the desired behaviour. The number of hidden nodes has been arbitrarily chosen as 3.

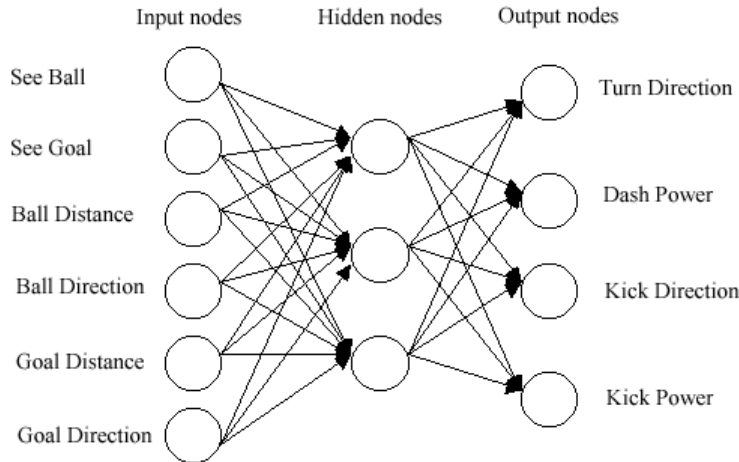


Figure 1: Neuro-player Architecture

### 4 Training Algorithm

The same algorithm as that used in the shock ice hockey work(Blair & Sklar 1998, Blair & Sklar 1999). A champion network is challenged by a series of mutant networks, until one is found that defeats the champion in a number of games. The champion's weights are then adjusted towards the mutant. The champion's weights are initialized to 0.0 and the steps for evaluating a generation are as follows:

1. Mutant  $\leftarrow$  Champion + Gaussian noise with standard deviation  $\sigma$  for each weight.
2. Champ and mutant play up to  $n$  games.
3. If mutant does better than the champ for most of the games played, then for each weight:

$$champ \leftarrow (1 - \alpha) * champ + \alpha * mutant \quad (0 \leq \alpha \leq 1)$$

The value of  $\alpha$  determines how much like the mutant the champ will become. A value of 1 replaces the champ with the mutant, a value of 0 leaves the champ unchanged. We have used  $alpha = 0.5$ .

A game consists placing the champion and the ball on the field, letting the clock run for  $s$  seconds, and recording the fitness of the player at the end of the game. The winner is the player that has the highest fitness. In the case of goal-only fitness this corresponds to the player kicking the highest number of goals.

### 5 Fitness Evaluation

Due to the randomness in the RoboCup environment, the same neuro-player having the same starting situation every time, performs differently each time. Thus, several games have to be played to avoid problems due to good and bad luck. In this implementation, a mutant must win at least 50% of the games to defeat the champion.

#### 5.1 Goal-Only Fitness Function

This is very straightforward – a player that kicks more goals in a game is fitter. However, there is no way differentiating between two players that do not score a goal, even though one of them may not have moved and the other one kicked the ball almost to the goal. Composite fitness is intended to deal with this situation.

## 5.2 Composite Fitness Function

The intention of using a composite fitness function is to be able to reward a player for partial achievement. This is particularly useful early in the evolutionary process when the players are particularly incompetent. The basic idea is determine the desirable factors in the player behaviour and combine them into a single numeric value with some kind of weighting scheme.

For a robocup dribble-and-score behaviour, factors such as seeing the ball, being near the goal and kicking the ball appear to be important:

To get composite fitness functions for our experiments we have selected factors from the following list and combined them using a scheme suggested in (Andre & Teller 1999). In this scheme the fitness is represented by a fixed point number such as

$$g.ddkknnntt$$

where  $g$  could be goals scored,  $dd$  being near the goal,  $kk$  kicking the ball,  $nn$  getting near the ball and  $tt$  seeing the ball.

**Player near the ball ( $nn$ )** These two digits are computed from scaling the distance between the player and the ball as received in the percept vector.

**Kicking the ball ( $kk$ )** This is incremented when the player kicks when the ball is less than 1.0 units away. It is decremented if the player kicks when the ball is not range and if the player does not kick if the ball is within range. It is evaluated only when the player sees the ball.

**Distance between ball and goal ( $dd$ )** These two digits are computed by using basic trigonometry on the data available from the percept vector and scaling the result to two digits. It is only computed if the player can see the ball and the goal.

**See ball and goal ( $ss$ )** This factor is incremented if the player can see both the ball and the goal. Otherwise it is decremented.

**See ball ( $tt$ )** This factor is incremented if the player can see the ball, otherwise it is decremented.

**See goal ( $hh$ )** This factor is incremented if the player can see the ball, otherwise it is decremented.

**Goal ( $gg$ )** Once a goal is scored, the game ends and a value of 1.0 is added to the final fitness value. The value is decremented for an own goal.

Figure 2 shows the fitness value changes after a particular sequence of events occurs. The fitness function used is  $g.hhttnn$ .

1. None of the fitness factors are satisfied. Fitness is 0.000000.
2. Player turns in the direction of the ball and sees it. Fitness is 0.000100.
3. Player moves towards the ball, getting near to it. Fitness is 0.000201.
4. Player turns in the direction of the goal, seeing both the goal and the ball. Fitness is 0.010302.

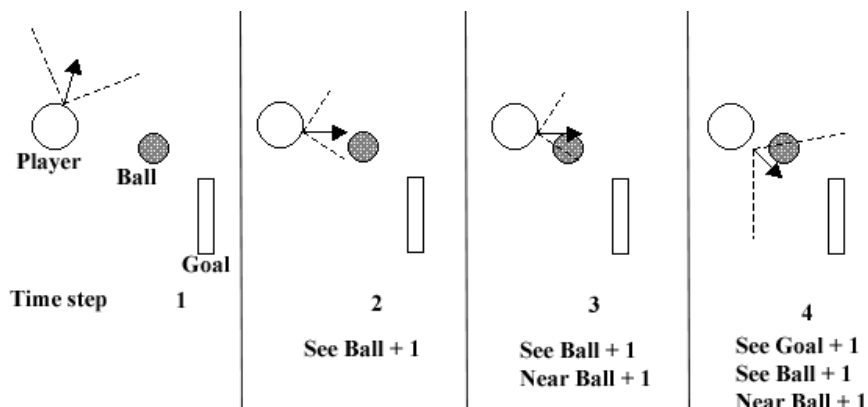


Figure 2: Neuro-player Fitness Evaluation.

## 6 Test Cases

To determine how well a trained player is performing and to be able to compare players trained in different ways we have constructed 200 test cases in which the ball and the player are in known fixed positions in the highlighted area of figure 5a. In the first set of 100 test cases the ball is near the goal and the player is at a distance of 1.0 units from the ball (meaning that the player is already close enough to kick the ball) and in the second set of 100 cases the player is at a distance of 5.0 units from the ball (meaning that the player first has to find and run to the ball). A game stops when a goal is kicked or when times run out after 10 seconds, thus the maximum score a player can get is 100 on each set of 100 test cases.

Figures 3c-6c show the performance of evolved players on these test cases. The players were tested towards the end of the evolutionary run. Also given is a qualitative description of the player behaviour.

## 7 Experiments and Results

Preliminary testing showed that  $\sigma = 1.0$ ,  $\alpha = 0.5$  and  $n = 10$  and  $s = 30$  seconds worked reasonably well. These values were used in all the experiments. Training is computationally very expensive – about 24 hours for 300 generations.

### 7.1 Experiment 1: Ball in center of field, player close to the ball and facing the goal, composite fitness function

In this experiment the objective is to determine whether a player can be trained from a seemingly straightforward situation: The ball is in the center of the field, the player does not need to run to the ball but is already close enough to kick it, the player is facing the ball and the goal, forming a straight line as shown in figure 3a. Such positioning allows the player to always see the ball and the goal at the start of the game, therefore biasing the *see ball and goal* and *distance between ball and goal* factors in the fitness function. The fitness function used was *g.ssddkkpp*. This fitness function is expected to encourage a behaviour in which the player always faces the ball and the goal, kicks towards the goal and keeps close to the ball.

Figure 3b shows the fitness value of the neuro-player per generation during training up to 1000 generations. The oscillations in fitness are probably due to the random factor added by the simulator to player commands. From the graph we see a rise in the fitness value by the 100th generation which is maintained until the 1000th generation, with some spikes. The ideal graph would be a gradual improvement in the fitness value over the generations during training. Even though figure 3a is not ideal, it does suggest that the algorithm improves the network performance upon training over 1000 generations.

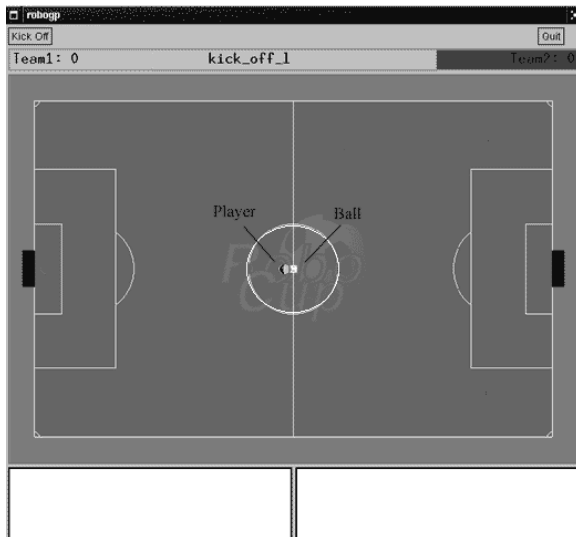
Figure 3c shows the behaviour of the player on the test set at generations 700, 800, 900 and 1000. The performance is reasonable (41 out of a possible 100 goals at generation 1,000) when the player begins close enough to the ball to kick it (distance to ball 1.0) and poor (0 out of a possible 100 goals) when it must find the ball first (distance to ball 5.0). However, the player is not totally incompetent, it is showing signs of developing the expected behaviour.

This experiment revealed some problems with the *see ball and goal* (*ss*) and *distance between ball and goal* (*dd*) factors. The *ss* factor encouraged the player to maintain a fixed distance from the ball, so as to see the ball and the goal at all times. The *dd* factor was very sensitive to the randomness in the robocup environment. When the ball was not moving different distances were returned in consecutive time steps. These two factors were thus removed from subsequent experiments.

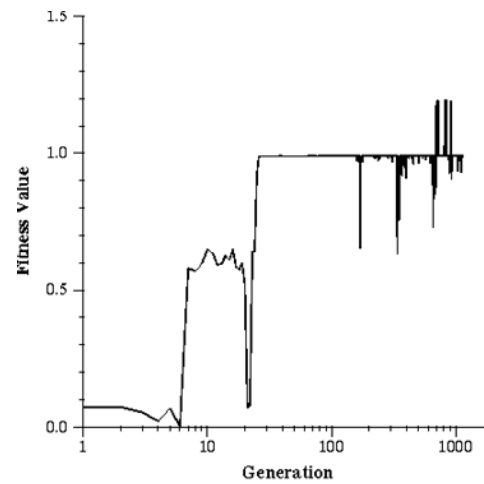
### 7.2 Experiment 2: Ball close to the goal, player in three different positions near the ball, player can see the goal, composite fitness

This experiment examines a slightly wider range of starting positions than experiment 1. The ball always starts near the goal with the player starting in 3 different positions at distance of 5.0 from it as shown in figure 4. The player must run to the ball first before a kick is possible. Such positioning biases the *see ball* factor *tt* in the fitness function. The fitness function used was *g.tttnnkkk*.

The fitness function is intended to develop a player that is always looking at the ball, keeping a close distance to it and kicking it towards the goal. The fitness graph shows considerably more oscillation than the previous experiment, however there is a slight suggestion of an improving trend after 500 generations, as the fluctuations in fitness are smaller. This player performed slightly better on both sets of test cases.



(a) Starting positions



(b) Fitness

Gen	Distance to ball is 1.0		Distance to ball is 5.0	
	Score	Description	Score	Description
700	30	The player either dribbles the ball, kicks it hard, maintains a distance from the ball or keeps still after kicking it. Actions can be towards the ball or in other directions	7	Player sometimes dashes towards the ball and kicks it in either the wrong direction or into the goal
800	40	As for 700	0	As for 700
900	28	As for 700	8	As for 700
1000	41	As for 700	0	As for 700

(c) Test Results

Figure 3: Experiment 1.

### 7.3 Experiment 3: Ball is in a random position near the goal, player is near the ball but may or may not be facing ball/goal, composite fitness

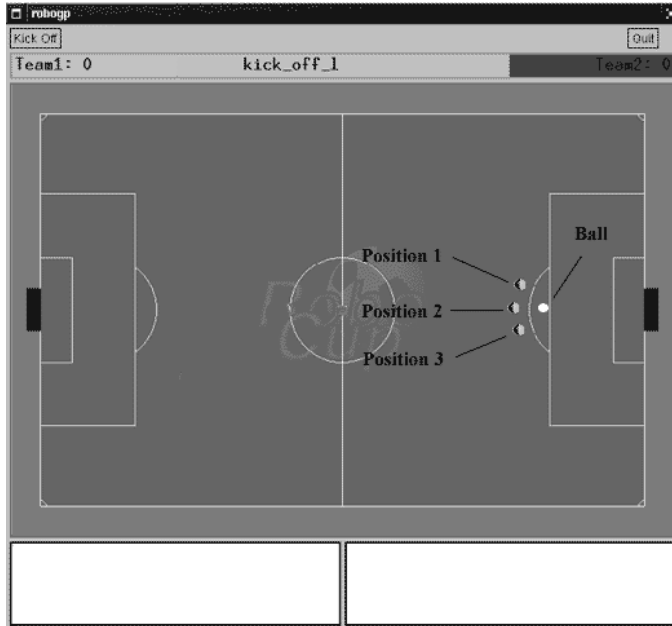
The aim of this experiment is determine whether the player can learn goal scoring behaviour if the ball starts anywhere within the shaded area near the goal (figure 5a) and the player within 1.0 units of the ball. The fitness function was the same as that used in experiment 2 (*g.tttnnkkk*).

Figure 5b shows the fitness value of the neuro-player per generation during training up to 1000 generations. Again there is significant oscillation, but there does not appear to be an improving trend. Performance on the test data is not as good as in either of the previous experiments.

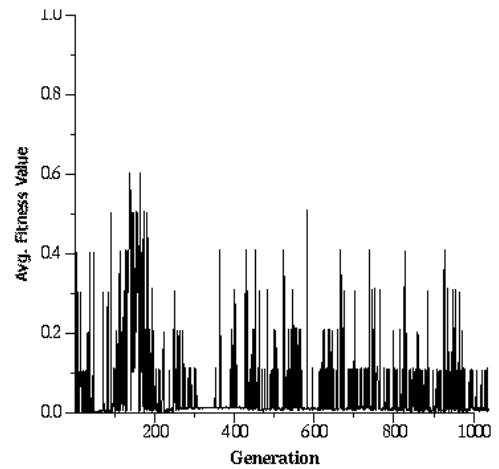
Figure 5c shows that the neuro-player has developed undesired behaviour of moving itself or the ball towards its own side of the field. With respect to the fitness function, the ideal case scenario would be for the player to see the ball at all times, go towards the ball and kick it into the goal. However, as stated above, the player evolved to a state where it just stared at the ball, either kicking the ball towards its own side of the field if it faced that way, or just moving back while keeping the ball in view. This is an example of the kind of exploitation of the fitness function stated as one of the main drawbacks described in (Blair & Sklar 1998) where the “The shaping process may introduce perverse incentives distracting a player from its main objective”.

### 7.4 Experiment 4: Ball is in a random position near the goal, player is near the ball but may or may not be facing ball/goal, goal-only fitness

This experiment was the same as experiment 3 except that goal-only fitness was used. Figure 6c shows the fitness value of the neuro-player per generation during training up to 1500 generations. Despite the large oscillations there is a definite trend of improvement. The performance on the first set of cases was the best of all four experiments This was in contrast to experiment 3 which was the same experiment with a composite fitness



(a) Starting positions



(b) Fitness

Gen	Distance to ball is 1.0		Distance to ball is 5.0	
	Score	Description	Score	Description
700	6	Most of the time the player kicks the ball towards its own side of the field or moves backwards towards its own side of the field	0	Most of the time the player kicks the ball towards its own side of the field or moves backwards towards its own side of the field
800	13	As for 700	0	As for 700
900	2	As for 700	8	As for 700
1000	4	As for 700	0	As for 700

(c) Test Results

Figure 4: Experiment 2.

function.

## 7.5 Effect of Added Randomness to Player Actions

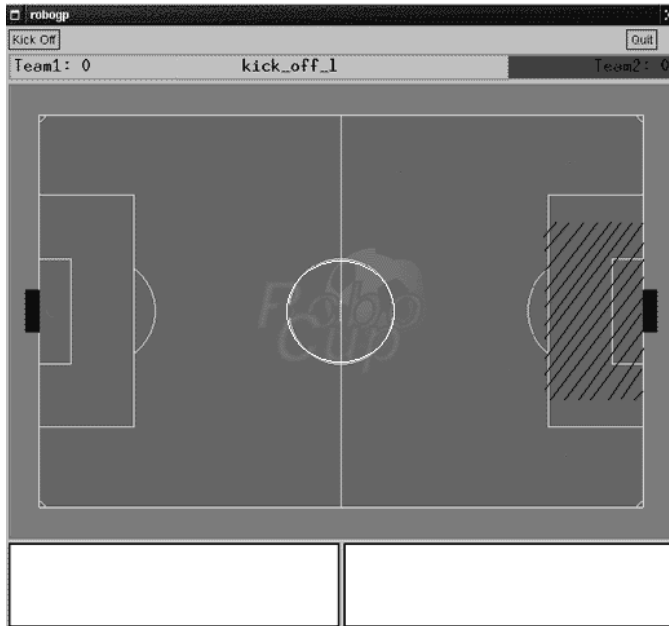
As described earlier, the soccer server adds randomness to the players' actions. To determine the effect of the randomness on the performance of the trained neuro-player we performed some additional tests on the player evolving during experiment 1. After each hundredth generation of training, the neuro-player was tested on a subset of 10 test cases, with and without the randomness. Table 2 shows a summary of these tests.

It is clear that the performance without randomness is much better than with randomness and that the this random factor has considerably increased the difficulty of training.

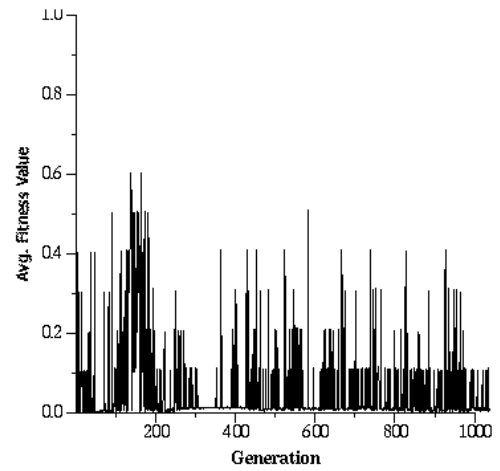
## 8 Conclusions

While some of the evolved players exhibited rudimentary 'dribble and score' skills, the overall results are somewhat disappointing considering the lengths of the runs. Based on the Shock experience considerably more computation time is needed to determine whether it will be possible to get competent players with the current network and training algorithm. It may well be that more complex networks with more input variables will lead to the evolution of better players more quickly.

Early indications are that the strategy of starting with simplified initial conditions and working up to more general, more complex ones will be successful. It also appears that goal-only fitness will be better. We expended a considerable amount of time designing, coding and testing composite fitness functions, only to see them not



(a) Starting positions



(b) Fitness

	Distance to ball is 1.0		Distance to ball is 5.0	
Gen	Score	Description	Score	Description
400	44	Kicks the ball straight ahead and turns round and round.	0	Player dashes and turns round and round
500	49	As for 400	0	As for 400
600	59	Kicks the ball straight and follows it	3	As for 400
7000	53	As for 400	0	As for 400

(c) Test Results

Figure 5: Experiment 3.

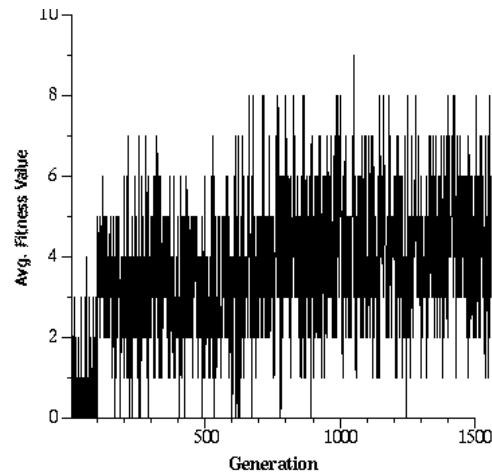
work quite as we expected. In the future we plan to use goal-only fitness with starting conditions of increasing levels of complexity, moving to the next level only when the player is competent at the current level.

## References

- Andre, D. & Teller, A. (1999). "Evolving Team Darwin United" In M. Asada & H. Kitano (eds), *RoboCup-98: Robot Soccer World Cup II*. Vol. 1604 of LNCS Springer Verlag Paris, France pp. 346–351.  
[\\*http://www.cs.cmu.edu/afs/cs/usr/astro/public/papers/Teller\\_Astro.ps](http://www.cs.cmu.edu/afs/cs/usr/astro/public/papers/Teller_Astro.ps)
- Blair, A. D. & Sklar, E. (1999). "Exploring Evolutionary Learning in a Simulated Hockey Environment" In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao & A. Zalzal (eds), *Proceedings of the Congress on Evolutionary Computation*. Vol. 1 IEEE Press Mayflower Hotel, Washington D.C., USA pp. 197–203.
- Blair, A. & Sklar, E. (1998). "The evolution of subtle manoeuvres in simulated hockey" *Proc. of SAB-5..*

	No Randomness	Randomness		No Randomness	Randomness
Gen	Score /10	Score /10	Gen	Score /10	Score /10
100	0	0	600	10	1
200	10	1	700	10	1
300	10	0	800	0	0
400	10	2	900	10	1
500	10	0	1000	0	0

Table 2: Effect of randomness



(a) Fitness

	Distance to ball is 1.0		Distance to ball is 5.0	
Gen	Score	Description	Score	Description
1200	62	Kicks straight most of the time. At some positions does not kick hard enough to score a goal. Sometimes kicks in wrong direction	0	Keeps turning round and round
1300	50	As for 1200	0	As for 1200
1400	54	As for 1200	0	As for 1200
1500	56	As for 1200	0	As for 1200

(b) Results on Test Cases

Figure 6: Experiment 4

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I. & Osawa, E. (1997). "RoboCup: The Robot World Cup Initiative" In W. L. Johnson & B. Hayes-Roth (eds), *Proceedings of the 1st International Conference on Autonomous Agents*. ACM Press New York pp. 340–347.

Luke, S. (1998). "Genetic Programming Produced Competitive Soccer Softbot Teams for RoboCup97" In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba & R. Riolo (eds), *Genetic Programming 1998: Proceedings of the Third Annual Conference*. Morgan Kaufmann University of Wisconsin, Madison, Wisconsin, USA pp. 214–222.  
 \*<http://www.cs.umd.edu/users/seanl/papers/robocupgp98.ps.gz>

Noda, I. (1998). *Soccer Server Manual*.  
<ftp://ci.etl.go.jp/pub/soccer/server/manual.r2.00.ps.gz>.

Noda, I., Matsubara, H., Hiraki, K. & Frank, I. (1998). "Soccer Server: a tool for research on multi-agent systems" *Applied Artificial Intelligence*. **12**(2-3).