Fifth Conference on Simulation of Adaptive Behavior, Zurich, 1998, 280-288

The evolution of subtle manoeuvres in simulated hockey

Alan D. Blair* **& Elizabeth Sklar****

* Dept. of Computer Science & Electrical Engineering
University of Queensland
4072, Australia
blair@cs.uq.edu.au

** Dept. of Computer Science
Brandeis University
Waltham, MA 02254, USA
sklar@cs.brandeis.edu

Abstract

A simulated hockey environment is introduced as a test bed for studying adaptive behavior and evolution of robot controllers. A near-frictionless playing surface is employed, partially mimicking zero gravity conditions. We show how a neural network using a simple evolutionary algorithm can develop nimble strategies for moving about the rink and scoring goals quickly and effectively.

1. Introduction

In recent years, there has been growing interest in using machine learning and evolutionary techniques for developing software controllers that play competitive or cooperative games in some kind of physical environment, either real or simulated. These have included: avoiding obstacles (Xiao et al., 1997; Lee et al., 1996; Brooks, 1986), foraging for "food" (Werger and Mataric, 1996), playing pursuit/evasion games (Miller and Cliff, 1994), discriminating objects (Beer, 1996), fighting for control of a cube (Sims, 1995) and RoboCup soccer (Kitano et al., 1995).

Following in these traditions, we have developed a simulated hockey game called *Shock* which is rather like "table-top" or "air" hockey. One or more players engage in games with the aim of shooting a puck into the opposing goal during an allocated time period. One player may be controlled by a human user; all other players are controlled by the simulation software.

Shock is similar in style to robot soccer simulators (Kitano et al., 1995; Salustowicz et al., 1998) and has much in common with (Balch, 1997), also being implemented in Java. However, Shock differs from previous systems in some significant ways. The near-frictionless environment in which Shock players operate provides special challenges distinct from those of typical playing surfaces, and the compactness of the rink distinguishes it from games played on an open plane (Miller and Cliff, 1994; Funes et al., 1998). While most work in robotics is concerned with collision *avoidance*, the Shock environment naturally gives rise to a strategy of collision *management*. Indeed, it is often advantageous to make strategic use of collisions in order to arrive at a desired position or orientation. The Shock players are rectangular in shape affording development of strategies involving internal movement and co-ordination different from those of circular players. In addition, Shock players are equipped with two-dimensional *holonomic* actuators which enable them to move with more grace and dexterity than standard one-dimensional wheels would allow.

We note that the near-frictionless Shock environment mimics to some extent the conditions found in outer space. Lately, hardware designs that will sustain a robot operating in zero gravity have been investigated. The Zero-G robot (Raibert et al., 1989) was designed to operate in a weightless environment such as a space station, and relies on a system of parallel rebound surfaces in order to control its movement. In work at CMU, the Self Mobile Space Manipulator (SM2) is being used to develop basic locomotion and manipulation capabilities for use on the international Space Station Freedom, with the ultimate goal of assisting astronauts during extravehicular activity and replacing astronauts in performing simple, dangerous or routine tasks (Nechyba and Xu, 1994).

2. The Simulated Hockey Domain

Shock is played in a rectangular rink, 1m wide by 1.5m long, with a goal at each end 150mm wide (Figure 1). Each player has a rectangular body 150mm by 50mm and a mass of 500 grams. The puck is circular in shape with a radius of 25mm and a mass of 100 grams.

Collisions between the players, puck and walls are calculated by the "spring" method of collision handling (Keller et al., 1993) and are totally elastic. This means that each experiences a restoring force in the normal direction, proportional to the depth of penetration. The

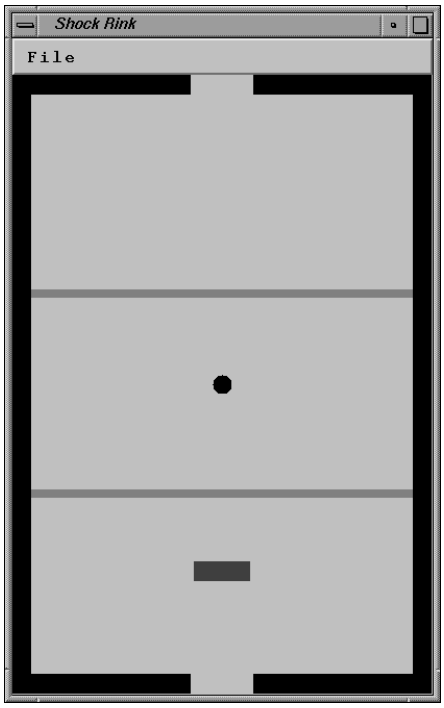

Figure 1 The Shock Rink.

puck collides frictionlessly with both players and walls, and therefore never acquires any spin. Each player experiences sliding friction when it collides with another player or a wall (i.e. a force in the opposite direction to the relative tangential motion and proportional to the restoring force). The players and puck also experience a frictional force as they slide on the rink that is proportional to their velocity and in the opposite direction.

We imagine each player as having a *skate* at either end of its body with which it can push on the rink in any direction (Figure 2). Actuators allowing 2-dimensional opposing movement in this way are difficult to implement in real hardware, but for the purpose of simulation we felt such a design might provide more interesting opportunities to study adaptive behavior.

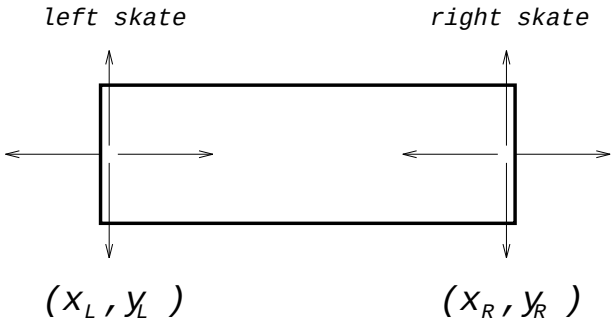

Figure 2 The Shock Player.

The output of the simulator's controller is in the form (x_L, y_L, x_R, y_R) where (x_L, y_L) and (x_R, y_R) represent the forces exerted at the left and right skate, respectively. We impose a restriction on the magnitude of the applied forces as follows:

$$x_L^2 + y_L^2 + x_R^2 + y_R^2 < \mathrm{F}_{\max}^2,$$

where $\mathrm{F}_{\max} = 0.2\mathrm{N}$. There is some redundancy in the format of the output since each player really only has three degrees of freedom available to it.

Each player is given complete information expressed in global coordinates, in the spirit of robot soccer where an overhead camera is often positioned above the playing field. In future work, we plan to develop players who instead receive their input from simulated local sensors.

The *game initial condition* (GIC) is a vector specifying a starting location and orientation for each player, and a starting location for the puck. All values are chosen at random, with the restriction that each player begins somewhere in the third of the rink nearest its own goal (see Figure 1). If only one player is present, the puck begins somewhere in the remaining two thirds of the rink; otherwise it begins somewhere in the middle third.

The simulation is updated in time intervals of 0.01 seconds, which runs in approximately real-time when the graphics window is open. When the graphics are switched off, the simulation runs faster, completing a game of 10 simulated seconds' duration in about half a second. Graphics mode is provided for several reasons: (1) to illustrate that the physics we are simulating appears natural, (2) to allow us to peek at the progress of the evolving players and (3) to provide an interface for the human player.

3. Network Architecture

Shock was designed so that players could be controlled by an expert system, fuzzy controller or any other kind of intelligent software. One of our objectives in building the system was to create an environment that could be used for comparing various machine learning techniques. The present work focuses on the following architectures:

1. simple recurrent (Elman, 1990) neural networks with 4, 6 and 8 hidden units
2. feed-forward networks with 6 and 8 hidden units
3. linear controller

All these architectures use 12 inputs and 4 outputs. Two of the inputs specify the location of the puck in a global co-ordinate system for which the origin $(0, 0)$ is at the center of the rink and the width of the rink is exactly 2.0 units. Four additional inputs indicate the locations of the player's left and right skate in the same co-ordinate system. The other 6 inputs convey the *observed velocity* of the puck and the left and right skate – i.e. the difference between current position and previous position, divided by the time interval.

The networks are fully connected and use a hyperbolic tangent function at the hidden layer. The four-dimensional output $\mathbf{z}$ of the network is converted to an applied force vector as follows:

$$(x_L, y_L, x_R, y_R) = \mathrm{F}_{\max} \tanh\left(||\mathbf{z}||\right) \frac{\mathbf{z}}{||\mathbf{z}||}.$$

This preserves the direction of the output vector, altering only its magnitude, and also allows the network to sometimes apply a very small force by choosing (x_L, y_L) and (x_R, y_R) to be nearly opposites of each other.

4. Evolutionary Paradigm

One long-term aim is to *co-evolve* different players to play games with each other lasting one minute or more of simulated time. However, before beginning multi-player simulations, we need to determine an architecture and a learning paradigm that will enable individual players to develop the basic skills necessary for the task. We therefore begin by studying a single-player 10-second version of Shock in which one player, beginning from a random GIC, moves about the rink until either (a) the puck goes into one of the goals or (b) the allotted time of 10 seconds elapses. In each game, a score of $+1$ is awarded for hitting the puck into the enemy goal, -1 for hitting it into the *friendly* goal (i.e. the player's own goal) and 0 for failing to hit it into either goal. Note that, even with no opponent, the need to avoid hitting the puck into the friendly goal is an important part of the task, since it prevents the player from winning by simply hitting the puck around randomly until a goal is scored.

We use an evolutionary hill-climbing algorithm in which a *champ* neural network plays two games with random GIC's and is then challenged by a series of *mutant* networks until one is found to do "better" (see below) than the champ; the champ's weights are then adjusted in the direction of the mutant:

1. mutant $\leftarrow$ champ + gaussian noise[1]
2. mutant plays 2 games with same GIC's as champ
3. if mutant does better than champ,
   champ $\leftarrow 0.9 *$ champ $+ 0.1 *$ mutant

If 100 mutants fail to do better than the champ with these GIC's, they are discarded and 2 new GIC's are generated. After the champ's weights have been adjusted, two new GIC's are chosen for the new champ and the process continues. The practice of making only a small adjustment in the direction of the mutant was introduced in previous work on backgammon (Pollack and Blair, 1998) on the assumption that most of the strategies of the (well-tested) champion would be preserved, with only limited influence from the mutant – since two games are not enough to determine for sure whether the mutant is really better, or just a lucky novice.

[1] with standard deviation of 0.05 for each weight

A total of 8 preliminary runs were conducted in order to refine various features of this algorithm. In the first five preliminary runs, only one game was used to compare the mutant and the champ, but the resulting players tended to degrade in performance after initial improvement – indicating that one game is not enough to make a good comparison.

Exposing the player to a variety of initial conditions is important for developing robust strategies (Tesauro, 1992). In the first two preliminary runs, the puck was constrained to always start in the middle third of the rink. The evolved players generally learned to hit the puck near to the enemy goal, but then failed to move up and hit the puck again – thus missing an opportunity to score. In subsequent runs, the puck was allowed to start anywhere in the upper two-thirds of the rink, which seems to provide an advantage because it enables players to learn the task "backwards from the end game" as is common in reinforcement learning (Tesauro, 1992).

What does it mean to do "better" than the champ? In theory, it should mean that the mutant is more likely to score goals than the champ. However, in the early stages of evolution the players are very unlikely to score at all, so it may become necessary to jump start the process. One way to do this is by introducing *partial credit* adjustments in the fitness measure which reward the player for moving closer to the puck, or for moving the puck closer to the enemy goal and away from the friendly goal. On the other hand, such adjustments run the risk of introducing artifacts which may distract the player from its main task (Angeline and Pollack, 1992). For example, a player seeking partial credit might push the puck close to the goal without bothering to score, or it might hover close to the puck without touching it, for fear of moving it in the wrong direction.

We attempt to balance these considerations by dividing the evolution into two phases: Phase 1 is the partial credit phase in which the mutant is considered to be better than the champ in the following circumstances:

1. the mutant scores in enemy goal but not the champ
2. the champ scores in friendly goal but not the mutant
3. both score in enemy goal, but mutant does it faster
4. both score in friendly goal, but mutant does it slower
5. neither scores a goal, but mutant moves the puck to a place where the *distance ratio* is smaller [2]
6. neither player ever hits the puck, but mutant is closer to puck at the end of the game than the champ was

In Phase 2, the fitness is evaluated purely on the basis of goals scored. Several GIC's are tested until two are found for which the champ scores either 0 or -1 (i.e. fails to hit the puck into the enemy goal in the allotted

[2] We define the *distance ratio* to be $d_1/(d_1 + d_0)$ where d_1 and d_0 are the distance from the puck to the enemy goal and the friendly goal, respectively.

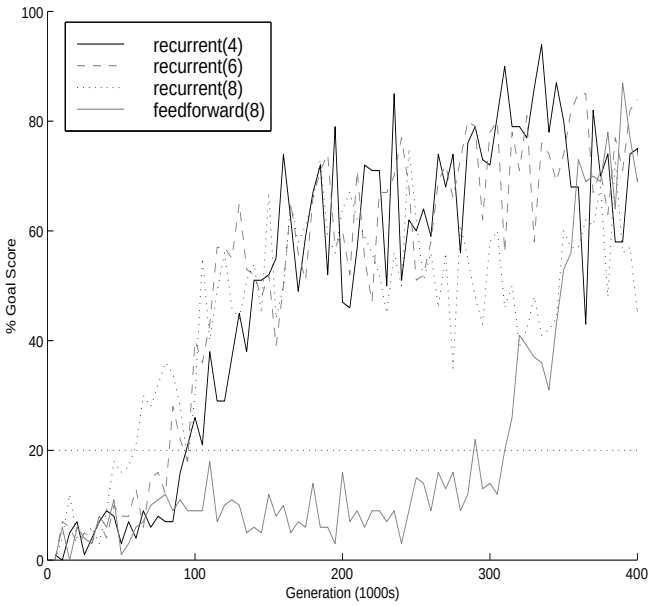

Figure 3 Score out of 100 games every 5000th generation for recurrent networks with 4, 6 and 8 hidden units and feed-forward network with 8 hidden units, annealed at 20%.

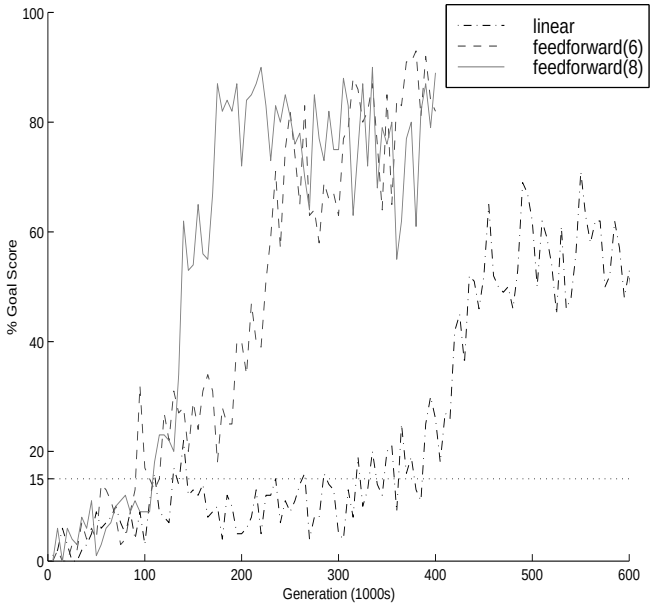

Figure 4 Score out of 100 games every 5000th generation for linear controller and feed-forward neural networks with 6 and 8 hidden units, annealed at 15%.

time). Then the mutant is considered to be better if it achieves a higher score than the champ for one of these GIC's, and an equal or higher score than the champ for the other GIC. Phase 2 was introduced because the eight preliminary runs, which used only the Phase 1 scheme, were found to produce inferior results.

5. Results

Following the 8 preliminary runs described in Section 4, a further 9 experimental runs were conducted in order to test the algorithm and compare different architectures and sizes for the neural network controller. Due to limited resources, each run was conducted once; ideally we would have executed each run several times and averaged the results.

In earlier work on phased learning (Pollack and Blair, 1998), we found that it is better to base the timing of the transition or *annealing* from one phase to another on internal diagnostics rather than at a pre-determined generation. To this end, we test our player every 5000 generations by having it play 100 games from different GIC's and using the total score for these games as a measure of its performance. When the performance according to this measure reaches some pre-determined *annealing threshold*, we switch from Phase 1 to Phase 2. The choice of an appropriate annealing threshold is a delicate and important issue, particularly when different architectures are being compared. If the threshold is set too low, causing the evolution to anneal prematurely, then the new evaluation criteria may prove so difficult that no mutant ever prevails. If it is set too high, the threshold may never be reached, denying us the potential benefits of subsequent phases. A threshold chosen for one kind of architecture may turn out to be inappropriate for another.

We initially chose 20% as our annealing threshold because all three recurrent networks reached this level within 100K (100,000) generations, after which their performance (without annealing) started to level off. With annealing implemented, their performance rose steadily to around 60% by generation 180K (Figure 3). On the other hand, the linear controller and feed-forward neural networks failed to reach this 20% threshold within 200K generations, which initially led us to suspect that these non-recurrent architectures might be inadequate for the task. However, when these runs were continued, the 8-unit feed-forward network finally reached the 20% level at generation 290K and, after annealing, its performance increased to 70% by generation 360K. We then repeated these runs with the annealing threshold set to 15% instead of 20% and found (Figure 4) that the 8-unit (respectively, 6-unit) network now annealed at 110K (resp. 95K) and reached 80% performance by 180K (resp. 370K). The linear controller annealed at 110K but took around 450K generations to reach 50% performance.

Figures 5 to 8 illustrate the steady advance in skill of the evolving recurrent network with 4 hidden units. The puck and player are shown in their initial configuration (solid) and at one-second intervals (outlined) while the dotted lines trace their trajectories throughout the game.

At generation 10K (Figure 5) the player approaches the puck but doesn't hit it. By generation 15K (not

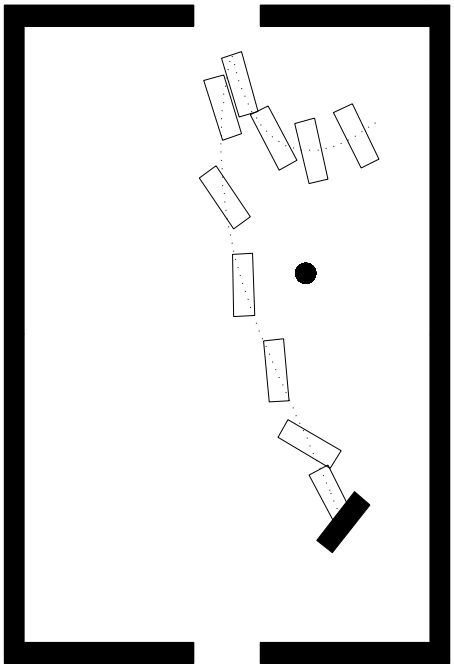

Figure 5 10K player, open shot.

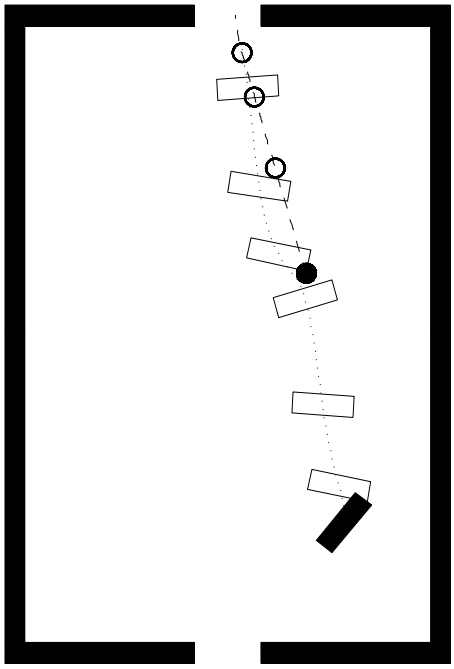

Figure 6 100K player, open shot.

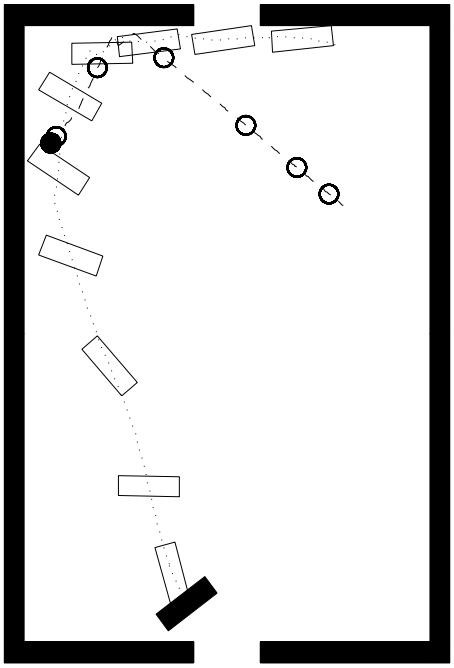

Figure 7 100K player, difficult shot.

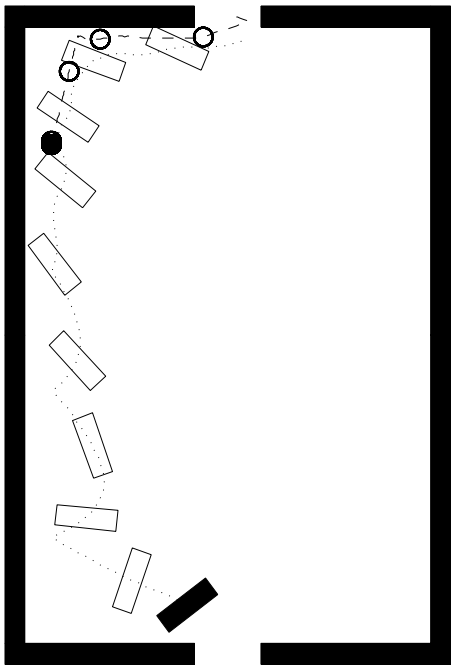

Figure 8 300K player, difficult shot.

shown) it can generally hit the puck once, not necessarily in the right direction. At generation 100K if the player has a fairly open shot at the goal (Figure 6) it hits the puck in the right direction and follows up in case a second shot should be required. If the same (100K) player is presented with a more challenging initial condition (Figure 7) it is able to nudge the puck away from the side wall, but upon hitting the back wall the puck fumbles from its grasp. By generation 300K the player has gained enough dexterity to see this situation through to a successful conclusion (Figure 8).

The other networks displayed similar kinds of behavior, but each developed its own particular style. The 6-unit recurrent network tends to maintain a greater distance from the puck, bouncing it once or twice off the back wall before scoring. Sometimes, when dribbling the

puck along the back wall as in Figure 8, it will overshoot the goal, but then chase the puck and start dribbling it back in the other direction to take a second shot. The 8-unit feed-forward network tends to stay very close to the puck, almost sliding it along the rink, and moves very slowly when it gets near to the goal. The linear controller hits the puck once, then backs up to the middle of the rink before approaching the puck again to take a second shot. It has difficulty scoring unless there is an open shot at the goal. This, together with its slow learning rate, suggest that a linear control mechanism may be inadequate for the task.

In further work, we hope to gain a better understanding of the relative strengths and weaknesses of the various types of networks, both on their own and against opponents.

6. Conclusions and Further Work

We have developed a simulated hockey environment and demonstrated how neural network controllers, learning to play on their own by means of an evolutionary algorithm, can develop reliable skills for scoring goals quickly and effectively. Our ultimate "goal" is to build teams of players for contests lasting a minute or more of simulated time.

We are currently applying co-evolutionary methods to this task, and also investigating on-line learning algorithms, including supervised and reinforcement learning, which may reduce the amount of computation currently required. Co-evolution and interaction with other players will undoubtedly bring a whole new set of complexities and challenges to this task (Salustowicz et al., 1998; Haynes et al., 1995). Nevertheless, given the nimble strategies we have been able to develop in the single-player scenario, it is not unreasonable to suppose that equally subtle manoeuvres appropriate for interaction with co-evolving opponents might be developed in the multi-player case.

In parallel work, we are building a user interface for an interactive version of Shock that will enable human players to test their skill against our various computer opponents. Check our web site for this coming attraction:

`http://www.demo.cs.brandeis.edu`

7. Acknowledgements

Thanks to Gordon Wyeth, David Blair, Janet Wiles and Jordan Pollack for helpful comments, and to Macquarie University for hosting the first author during the writing of this paper. This work was supported primarily by a University of Queensland Postdoctoral Fellowship.

References

Angeline, P. and Pollack, J. (1992). Evolutionary induction of subroutines. In *Proc. 14th Annual Conference of the Cognitive Science Society*, pages 236–241.

Balch, T. (1997). Learning roles: Behavioral diversity in robot teams. *AAAI97 Workshop on Multiagent Learning.*

Beer, R. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. In (Maes et al., 1996), pages 421–429.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Transactions on Robotics and Automation*, 2:14–23.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Funes, P., Sklar, E., Juillé, H., and Pollack, J. (1998). Animal-animat coevolution: Using the animal population as fitness function. In *Proceedings of SAB-5.*

Haynes, T., Sen, S., Schoenefeld, D., and Wainwright, R. (1995). Evolving a team. In *AAAI 1995 Fall Symposium on Genetic Programming.*

Keller, H., Stolz, H., Ziegler, A., and Bräunl, T. (1993). Virtual mechanics – simulation and animation of rigid body systems. Rept. Comp. Sci. Dept. 8/93, U. Stuttgart.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. (1995). Robocup: The robot world cup initiative. In *IJCAI-95 Workshop on Entertainment and AI/ALife.*

Lee, W.-P., Hallam, J., and Lund, H. (1996). A hybrid gp/ga approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks. In *Proc. 1996 IEEE Int'l Conf. Evolutionary Computation*, pages 384–389.

Maes, P., Mataric, M., Meyer, J.-A., Pollack, J., and Wilson, S., editors (1996). *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior.* MIT Press.

Miller, G. and Cliff, D. (1994). Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 411–420.

Nechyba, M. and Xu, Y. (1994). Sm2 for new space station: autonomous locomotion and teleoperation control. *Proc. IEEE Int'l Conf. Robotics and Automation*, 2:1765–1771.

Pollack, J. and Blair, A. (1998). Co-evolution in the successful learning of backgammon strategy. *Machine Learning (to appear).*

Raibert, M., H.B. Brown, J., Chepponis, M., Hodgins, J., Koechling, J., Dustman, D., Brennan, W. K., Barrett, D., Thompson, C., Hebert, J., Lee, W., and Borvansky, L. (1989). Dynamically stable legged locomotion. AI technical report 1179, MIT.

Salustowicz, R., Wiering, M., and Schmidhuber, J. (1998). Learning team strategies: soccer case studies. *Machine Learning (to appear).*

Sims, K. (1995). Evolving 3d morphology and behavior by competition. *Proc. Artificial Life 4*, pages 28–39.

Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.

Werger, B. and Mataric, M. (1996). Robotic "food" chains: Externalization of state and program for minimal-agent foraging. In (Maes et al., 1996), pages 625–634.

Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K. (1997). Adaptive evolutionary planner/navigator for mobile robots. *IEEE Trans. Evol. Computation*, 1(1):18–28.