Australian Journal of Intelligent Information Processing Systems **4**, 1997, 166–176

What makes a good co-evolutionary learning environment?

Alan D. Blair
Dept. of Computer Science
University of Queensland
4072, Australia
blair@cs.uq.edu.au

Jordan B. Pollack
Dept. of Computer Science
Brandeis University
Waltham, MA 02254
pollack@cs.brandeis.edu

Abstract

There is growing evidence to suggest that the success of a co-evolutionary learning system may depend critically on the nature of the environment in which the learner is placed, and on certain attributes of the task domain, rather than the details of the particular learning algorithm employed. We discuss how a learning system can be modeled as a meta-level game between abstract entities which we call *performer*, *infiltrator* and *evaluator*. Learning can sometimes fail due to collusive suboptimal equilibria in this *meta-game of learning*. But some domains have special attributes which seem to prevent such collusions and thereby facilitate co-evolutionary advancement. A better understanding of these issues may help to improve the design of co-evolutionary learning systems in the future.

1. Introduction

The success of a machine learning system depends very much on the learning environment in which it is placed. After it has extracted all the accessible information from its original environment, it may need to be put into a new, more challenging, environment in order to progress. 'Curricular' or 'staged' learning occurs when a learner is placed into a pre-designed series of environments one after the other, as it progresses (Langley, 1995). However, designing an appropriate series of environments may be very difficult. This difficulty would be avoided if there were some way for the learner and its environment to *co-evolve* with each other, so that the one would always be appropriate for the other.

Strategic games provide a good opportunity to study this kind of co-evolutionary learning. In theory, several machine learning systems trying to master a competitive game could all learn to improve their strategies simultaneously by playing each other and observing the outcomes – as each one improved, it would provide a slightly more challenging opponent for the others, fuelling a continuing spiral of advancement.

While this idea has been around since the early days of artificial intelligence (Samuel, 1959, Michie, 1961) most applications of it – for example, to chess, go, tic-tac-toe and other games – have run into serious difficulties. In order to understand these difficulties, we note that a learning system can, in general, be modeled as a meta-level game which we call the *meta-game of learning* or MGL (Pollack & Blair, 1998) and to which we can apply a game theory analysis. In some cases, learning may be stifled due to various kinds of *collusion*, which show up as suboptimal equilibria in this MGL, and indeed many of the problems encountered by co-evolutionary learners can be put down to collusions of this kind. One example is where the players repeatedly draw each other; another is a *narrowing of scope* in which they keep playing the same kinds of games over and over, only exploring some narrow portion of the strategy space, and missing out on key regions where they would then be vulnerable to humans or other players.

However, there have been a few notable cases in which these problems have apparently been avoided. One such instance came to light when Gerald Tesauro (1992) compared two different methods for training neural networks to play the game of backgammon. The first network was trained on a large database of hand-crafted positions, with corresponding moves chosen by a human expert; the second network was trained by having it play against itself thousands of times and using the outcome of each game to make a small adjustment in its strategy according to the *temporal difference* or *TD-learning* algorithm (Sutton, 1988). Surprisingly, the network that was trained by self-play, though it initially played a poor (essentially random) game, eventually surpassed the network trained on the expert database, and a later version called TD-Gammon (incorporating some additional hand-crafted features) achieved world master level play (Tesauro, 1995). While the success of TD-Gammon was originally attributed to the strength of the TD-learning algorithm, we have shown in previous work (Pollack & Blair, 1998) that results similar to (Tesauro, 1992) can be achieved using a simple hill-climbing algorithm in place of TD-learning. This 'HC-Gammon' result – along with the fact that TD-learning has not led to similar impressive breakthroughs in other, comparable, domains – suggests that a large measure of TD-Gammon's success might be attributed to the nature of the co-evolutionary learning framework, and to certain features of the backgammon domain itself which work to prevent collusion in the MGL. A better understanding of these issues may help to improve the design of co-evolutionary systems in the future.

In the present work, we provide a more detailed explanation of the MGL framework with respect to both evolutionary and co-evolutionary learning (Section 2). We then briefly review the HC-Gammon results (Section 3) and discuss how backgammon and other domains can be analysed within the MGL framework (Section 4).

2. Meta-Game of Learning

2.1 MGL Framework and Collusion. The familiar problem of premature convergence in machine learning or evolutionary algorithms can be looked at from a game theoretic perspective. The most obvious way for a candidate solution in a machine learning environment to increase its chance of survival is by improving its ability to perform the requisite task, i.e. raising its own fitness. But this is not the only way to survive. Another way is to pass on to *other* candidate solutions certain attributes which will *restrict their ability* to perform the task. This could be achieved in a genetic algorithm by passing on strategic genes to other members of the population, or in a hill-climbing or gradient-descent algorithm by guiding the search into a region of the space where a local optimum is located. Although it may seem paradoxical, in some cases considerable advantage can be gained by *infiltrating* others in this way, especially if the population is small or the effective dimensionality low. We can anthropomorphise these competing influences and think of the learning process as an interaction between a *performer*, which looks for opportunities to improve performance on the task, and an *infiltrator*, which looks for opportunities to restrict the performance of other candidate solutions. The infiltrator and performer interact with each other in a kind of meta-level game which we call the MGL.

In general, we hope that the performer and infiltrator will be working against each other and that the performer will have the upper hand. However, certain features of the task domain or training environment may introduce Nash equilibria into the MGL which allow the performer and infiltrator to *collude* with each other and drag the system into a suboptimal solution. Suppose the state space can somehow be partitioned into two subspaces X and Y – for example, X could denote the set of positions associated with a particular style of game, or those which follow a certain decisive move or sequence of moves. Then one strategy for the *infiltrator* in the MGL might be to move into the X region whenever possible. We will call this strategy choose(X). An alternative strategy would be choose(Y), and other strategies may exist which sometimes move into X, sometimes into Y. One strategy for the *performer* would be to *specialize* in X by introducing attributes which enhance performance in the X region (possibly to the detriment of the Y region). We will call this strategy specialize(X). An alternative strategy would be specialize(Y), and other strategies may exist which don't particularly specialize in either X or Y.

$$\begin{array}{ll} \text{performer:} & \text{specialize(X)} \\ & \downarrow\uparrow \\ \text{infiltrator:} & \text{choose(X)} \end{array}$$

Figure 1. Positive feedback between choose(X) and specialize(X) creates an opportunity for collusion in the MGL between the infiltrator and performer.

Now suppose that the performer follows the strategy specialize(X) while the infiltrator follows the strategy choose(X). Then the performer helps the infiltrator by creating a selective advantage for choosing X (see Figure 1) and the infiltrator helps the performer by spreading its influence to other candidate solutions and ensuring that they will also move into the X region whenever possible, thus *narrowing the scope* of the search and providing the performer with an ideal arena in which to showcase its own strengths. This collusion between the infiltrator and performer may force the algorithm to converge prematurely on a suboptimal solution (assuming fitter solutions are possible which move freely between X and Y).

2.2 Co-evolutionary MGL. From this game theoretic perspective, techniques to combat premature convergence such as fitness sharing, hall of fame, shared sampling, etc. (Rosin & Belew, 1997) can be seen as attempts to remove opportunities for collusion between the infiltrator and performer by changing the structure of the MGL. One such method is co-evolutionary learning, in which two learning systems A and B competitively learn by acting as opponents for each other (Hillis, 1992).

In addition to the performer and infiltrator roles, each system now plays a new role – namely, that of *evaluator* for candidate solutions of the other co-evolving system.[1] The evaluator role of the B system can work to prevent collusion in the A system, as illustrated in Figure 2. The infiltrator of A, by following the strategy choose$_A$(X), provides a selective advantage within the B system for the strategy specialize$_B$(X). But specialize$_B$(X) provides a selective **dis**advantage for choose$_A$(X), thus creating a negative feedback loop which may remove the opportunity for collusion between the infiltrator and performer of the A system.

But co-evolution is a two-edged sword. Although removing some opportunities for collusion, regrettably it can also create others. For example, suppose that **both** players participate in choosing between X and Y. Then a positive feedback loop can be established (Figure 3) from specialize$_A$(X) to choose$_A$(X) to specialize$_B$(X) to choose$_B$(X) and back to specialize$_A$(X). Depending on the structure of the domain,

[1]The evaluator and performer correspond roughly to what were called the *teacher* and *student* in (Pollack & Blair, 1998) and (Sklar, Blair & Pollack, 1998).

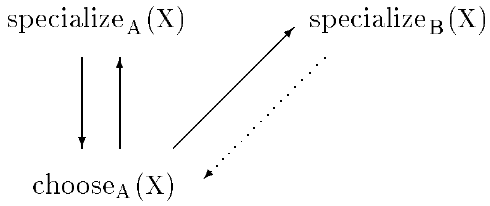


Figure 2. With co-evolution, the new role of evaluator creates a negative feedback loop between choose$_A$(X) and specialize$_B$(X), which may undermine collusion between the infiltrator and performer.

this positive feedback loop may be strong enough to override the negative influence from specialize$_B$(X) to choose$_A$(X) and vice-versa, leading to a new kind of collusive suboptimal equilibrium in this co-evolutionary MGL[2] (see also Figure 6).

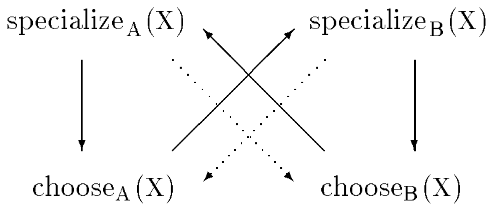


Figure 3. If both players participate in choosing X, a new positive feedback loop emerges connecting specialize$_A$(X) choose$_A$(X) specialize$_B$(X) and choose$_B$(X). This can create a new opportunity for collusion in the MGL involving the performer and evaluator roles of both the A and B learning systems.

3. HC-Gammon

In this section we briefly review the results of (Pollack & Blair, 1998) in which a hill-climbing algorithm was used to develop a neural network backgammon player (see Figure 4). A standard 2-layer feedforward neural network was set up in the same fashion as (Tesauro, 1992) with 4 input units to represent the number of each player's pieces on each of the 24 points, plus 2 inputs each to indicate how many are on the bar and off the board. In addition, one more unit was added which reports whether or not the game has reached the racing stage, making a total of 197 input units. These were fully connected to 20 hidden units, in turn connected to one output unit which judges the position. The game is played by generating all legal moves, converting the resulting board positions into the proper network input, and choosing the one judged as best by the network. We start with all weights set to zero. Our initial algorithm was simple hillclimbing:

1. add gaussian noise[3] to the weights of the network to create a mutant
2. play the network against the mutant for a number of games
3. if the mutant wins more than half of these games, select it for the next generation

Surprisingly, this worked reasonably well. The networks so evolved improved rapidly at first, but then sank into mediocrity. The problem we perceived is that comparing

[2]In the case of learning through self-play (as in TD-Gammon), A and B are consolidated and a new MGL emerges in which the roles of performer, infiltrator and evaluator interact in a more complex way. However, in the present work we will not make a distinction between co-evolutionary learning and self-learning.

[3]with standard deviation of 0.05

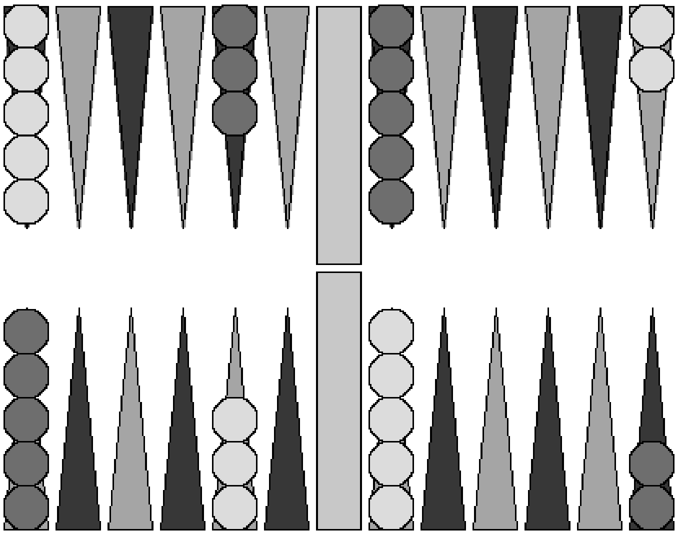


Figure 4. Backgammon is a board game which has been popular in the Middle East and in Europe for more than 2000 years. Two players take turns throwing dice and moving pieces around a board comprising 24 locations or *points*. Once a player has moved all their pieces to the last 6 of these points, they can begin moving them off the board. The first player to move all their pieces off the board is the winner. It is forbidden to move a piece to a point already occupied by two or more opponent pieces, but it is okay to move to a point occupied by only one opponent piece, in which case that piece will be placed on a strip in the center of the board (called the *bar*) and will have to start again from the beginning. The two players move their pieces in opposite directions. When all of one player's pieces have passed all of the other player's pieces, and there are no more opportunities to put opponent pieces on the bar, we say that the *contact* stage of the game has ended and the *racing* stage has begun.

two close backgammon players is like tossing a biased coin repeatedly: it may take dozens or even hundreds of games to find out for sure which one is better. Replacing a well-tested champion is dangerous without enough information to prove that the challenger is really a better player and not just a lucky novice. Rather than burden the system with so much computation, we instead introduced the following modifications to the algorithm to avoid this 'Buster Douglas Effect'[5]: Firstly, the games are played in pairs, with the order of play reversed and the same random seed used to generate the dice rolls for both games. This washes out some (though not all) of the unfairness due to the dice rolls when the two networks are very close. Secondly, when the challenger wins the contest, rather than just replacing the champion by the challenger, we instead make only a small adjustment in that direction:

$$\text{champion} \leftarrow 0.95 * \text{champion} + 0.05 * \text{challenger}$$

This idea, similar to the 'inertia' term in back-propagation (Rumelhart et al., 1986) was introduced on the assumption that small changes in weights would lead to small changes in decision-making by the evaluation function. So, by preserving most of the current champion's decisions, we would be less likely to have a catastrophic replacement of the champion by a lucky novice challenger. In the initial stages of evolution, two pairs of parallel games were played and the challenger was required to win 3 out of 4 of these games.

[5]Buster Douglas was world heavyweight boxing champion for nine months in 1990.

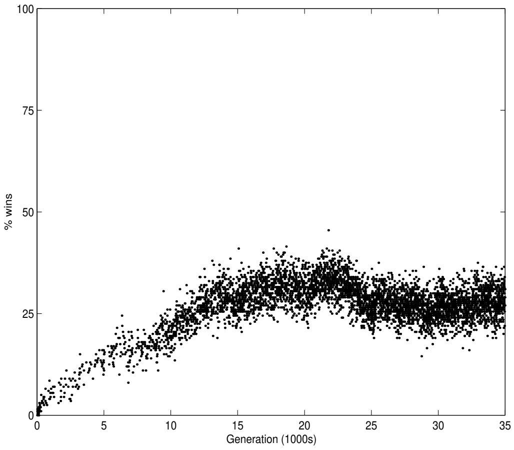


Figure 5(a). Algorithm without annealing. Percentage of wins (out of 200 games) of first 35,000 players against PUBEVAL.

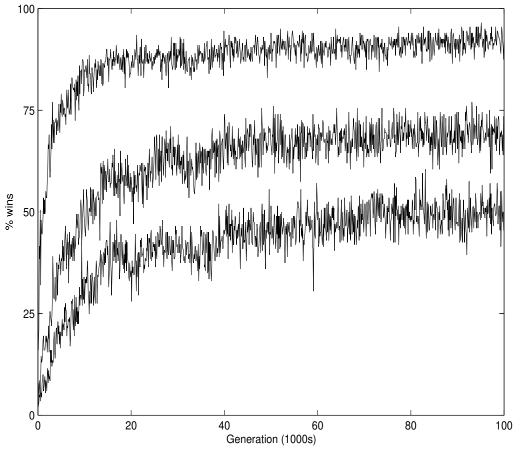


Figure 5(b). Annealed algorithm: Percentage of wins against benchmark networks 1k (upper), 10k (middle) and 100k (lower).

Figure 5(a) shows the first 35,000 generation players rated against a strong public-domain player called PUBEVAL, trained by Tesauro using human expert preferences. There are three things to note: (1) the percentage of wins against PUBEVAL increases from 0% to about 33% by 20,000 generations, (2) the frequency of successful challengers increases over time as the network improves, and (3) in some places (e.g. starting at 20,000) the performance against PUBEVAL begins to falter. The first fact shows that our simple self-playing hill-climber is capable of learning. The second fact is quite counter-intuitive – we expected that as the player improved, it would be harder to challenge it. This is true with respect to a uniform sampling of the 4000 dimensional weight space, but not true for a sampling *in the neighbourhood* of a given player: once the player is in a good part of weight space, small changes in weights can lead to mostly similar strategies, ones which make mostly the same moves in the same situations. However, because of the few games we were using to determine relative fitness, this increased rate of change allows the system to drift, which may account for the subsequent degrading of performance.

To counteract the drift, we decided to change the rules of engagement according to the following 'annealing schedule': after 10,000 generations, the number of games that the challenger is required to win was increased from 3 out of 4 to 5 out of 6; after 70,000 generations, it was further increased to 7 out of 8 (Note: the numbers 10,000 and 70,000 were originally chosen on an ad hoc basis from observing the frequency of successful challenges in this run; in replication experiments, the number of games was increased every time the challenger success rate exceeded 15% when averaged over 1,000 generations). After 100,000 games, this 'annealed' hill-climbing algorithm led to a surprisingly strong player, capable of winning 40% of the games against PUBEVAL.

In co-evolutionary learning, it is always important to test later generation players against earlier ones in order to see whether progress is monotonic, or punctuated with cycles of dominance followed by invasion (Axelrod, 1984). To this end, players were sampled every 100 generations and tested against three 'benchmark' networks at generation 1,000, 10,000 and 100,000. Figure 5(b) shows the percentage of wins for the sampled players against the three benchmark networks. Note that the three curves cross the 50% line at 1, 10 and 100, respectively and show a noisy but nearly monotonic increase in player skill as the evolution proceeds.

4. Discussion

These results show that the game of backgammon can be learned quite well with a co-evolutionary approach – whether it be a sophisticated technique like TD-Learning or a simple hillclimbing algorithm. We do not claim HC-Gammon to be as good as TD-Gammon, but it is surprisingly good considering the simplicity of the algorithm used – which strongly suggests that there is something special about the domain of backgammon which allows co-evolutionary learning techniques to succeed.

4.1 Common Features. TD-Learning has worked well on a number of other tasks including elevator control (Crites & Barto, 1996) and job shop scheduling (Zhang & Dietterich, 1996). In addition, co-evolutionary learning techniques have been successfully applied in a variety of areas. One example is the *Evolving Virtual Creatures* (EVC) domain: in a simulated competitive game devised by Karl Sims (1995) two virtual creatures in a flat playing arena compete for control of a cube placed initially between them. In each round of competition, all creatures from one species play against the champion of the other species from the previous round. Over several generations, competing species were observed to leap-frog each other in evolutionary arms races, as they each discovered methods for reaching the cube, and then further evolved strategies to counter the opponent's behaviour. Some creatures pushed their opponent away from the cube, some moved the cube away from its initial location and then followed it, while others simply covered up the cube to block the opponent's access.

We list here some of the features which these domains seem to have in common:

Ergodicity: The backgammon and EVC domains are both *ergodic* in the sense that any position can be followed at a later time by any other position.[6]

Continuity: The EVC domain makes use of a continuous state space with no discrete edges or internal boundaries. The same is true for a number of other simulated physics domains to which co-evolutionary learning techniques have been successfully applied (e.g. Miller & Cliff, 1994).

Stochasticity: The moves in backgammon are governed by random dice rolls, which leads self-play into a much larger part of the search space than would otherwise be explored, as noted in (Tesauro, 1992).

Broad Spectrum of Opportunity: In both of these domains each player apparently has available to it, at any given time, a number of avenues for improvement. In backgammon, there are many aspects of the game which can be developed independently (e.g. blocking, racing, back-game strategy, etc.) Virtual creatures, like their biological counterparts, can improve by developing a slightly longer arm, slightly better sensors, or becoming slightly faster, etc. This is in contrast to some other domains where learning can only proceed along a particular path (learn A, then B, then C, etc. in a pre-determined order).

4.2 MGL Revisited. The kind of collusion described in Section 2 relies on the ability to partition the state space into two regions X and Y. We hypothesise that in some domains, certain features including those listed above somehow preclude this kind of collusion, by preventing the state space from being so neatly partitioned. In the EVC domain, for example, our evaluator and performer might in theory try to forge an agreement whereby the evaluator always keeps the cube in the left hand

[6] With the exception, in backgammon, of racing positions that occur in the last few moves of the game, and contact positions with some pieces off the board or out of play, which have only a marginal impact on the playing strategy.

side of the arena, or within a certain distance of its starting position, or below a certain speed, etc., while the performer develops special skills to handle situations with those particular properties. But in fact the performer is unlikely to comply with any such agreement because the *continuity* of the domain allows parameters such as maximum speed or distance to be adjusted gradually in small steps until they cover the entire beneficial range. Competence in the left hand side of the arena can be gradually expanded to include competence in the center of the arena and eventually competence in the whole arena, because there is no 'natural barrier' to halt its expansion. Furthermore, ergodicity and stochasticity will make it difficult for the evaluator to keep its part of the bargain. If any state can potentially be followed at a later time by any other state, the task of restricting players to a proper subset of the state space cannot be achieved with a single decisive move but instead requires ongoing vigilance on the part of the evaluator.

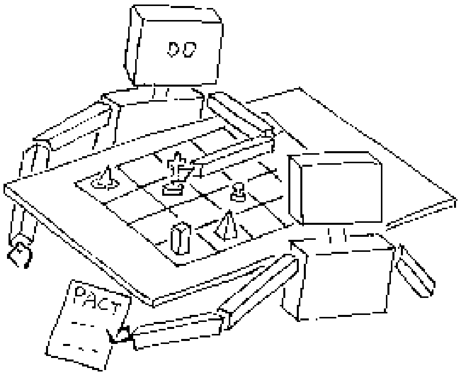


Figure 6. Collusion in the MGL may be thought of as a kind of *secret pact* between the players to always play a particular style of game, or to play for a draw rather than a win or loss, or to take turns giving the other player an easy win by making an early sequence of suboptimal moves.

Draws are impossible in backgammon so collusion by repeated draws is precluded. Some domains allow for a kind of collusion where each player takes turns giving the other player an easy win (Angeline, 1994) by making an early sequence of suboptimal moves. However the stochasticity of backgammon seems to make this unlikely because a position that appears to provide an easy win may in fact turn out to be a losing position depending on the outcome of subsequent dice rolls. What many observers find exciting about backgammon, and what helps a novice sometimes overcome an expert, is the number of situations where one dice roll, or an improbable sequence, can dramatically reverse which player is expected to win.

In order to quantify this 'reversibility' effect, we collected some statistics from games played by HC-Gammon (100k) against itself. For each n between 0 and 120 we collected 100 different games in which there was still contact at move n, and, for $n > 6$, 100 other games which had reached the racing stage by move n (but were still in progress). We then estimated the probability of winning from each of these 100 positions by playing out 200 different dice-streams. Figure 7 shows the standard deviation of this probability (assuming a mean of 0.5) as a function of n, as well as the probability of a game still being in the contact or racing stage at move n. Figure 8 shows the distribution in the probability of winning, as a function of move number, symmetrized and smoothed out by convolution with a gaussian function.

These data indicate that the probability of winning tends to hover near 50% in the early stages of the game, gradually moving out as play proceeds, but typically remaining within the range of about 15% to 85% as long as there is still contact, thus allowing a reasonable chance for a reversal. Our conjecture is that these dynamics facilitate the learning process by providing, in almost every situation, a nontrivial chance of winning and a nontrivial chance of losing, therefore *potential to learn from*

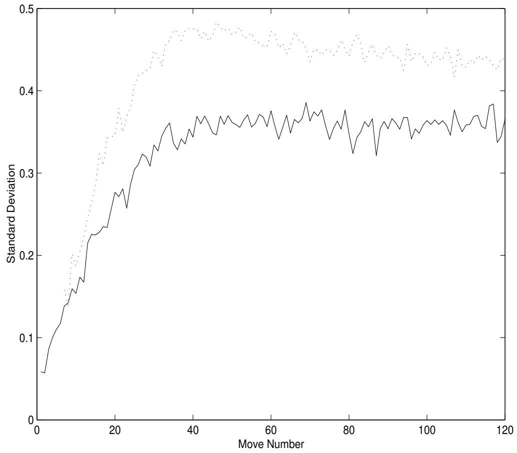


Figure 7(a). Standard deviation in the probability of winning for contact positions (solid) and racing positions (dotted).

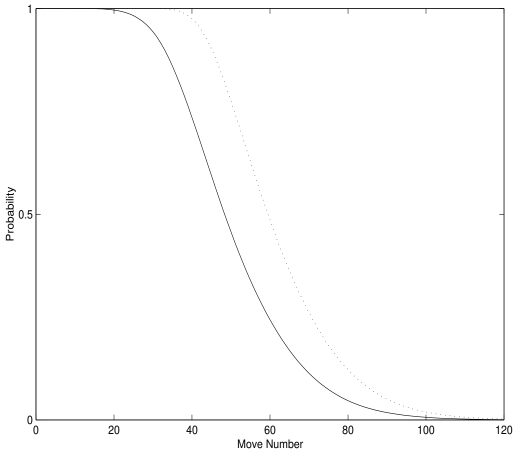
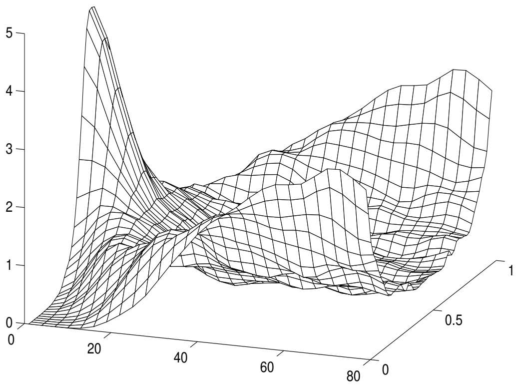


Figure 7(b). Probability of a game still being in the contact stage (solid) or racing stage (dotted) at move n.



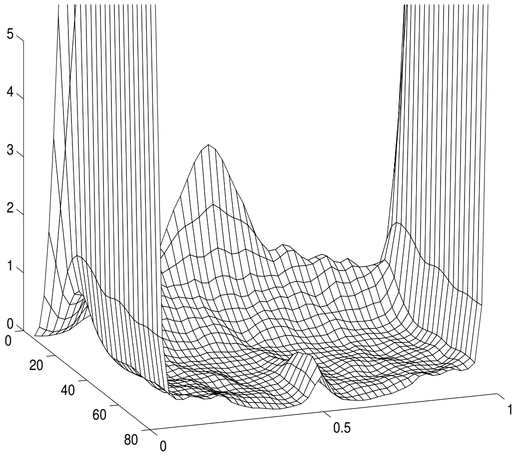


Figure 8. Smoothed distributions of the probability of winning as a function of move number, for contact positions (left) and racing positions (right).

the consequences of the current move. This is in deep contrast to many other domains in which early blunders could lead to a hopeless situation from which learning is virtually impossible because the reward has already become effectively unattainable.

Many other reinforcement learning tasks also involve stochasticity of one kind or another (Zhang & Dietterich, 1996, Crites & Barto, 1996). Additionally, many attempts have been made to add randomness or force initial moves in deterministic games (Fogel, 1993, Epstein, 1994, Walker et al., 1994, Schraudolph et al., 1994). However the mere addition of randomness cannot be expected to make the problem tractable in all cases. The critical factor is whether it makes enough difference in the structure of the MGL to remove opportunities for collusion.

5. Conclusion and Further Work

While the traditional focus in machine learning has been on the learning algorithms employed, the prevention of stagnation or premature convergence in evolutionary algorithms has also been an active area of research.

We believe much can also be learned by focusing on the nature of the environments and the tasks being learned. Indeed, certain domains appear to have special attributes which facilitate co-evolutionary advancement by preventing collusive suboptimal equilibria in the meta-game of learning. In ongoing work, we are trying to identify and study other domains with similar attributes, in the hope that a better understanding of these phenomena may lead to improved design of co-evolutionary learning systems in general.

Acknowledgments

Thanks to Elizabeth Sklar for helping to improve the presentation of this work, and to Gerald Tesauro, Tom Dietterich, Wei Zhang, Rich Sutton, Andrew Moore, Justin Boyan, Mark Pendrith, Xin Yao and David Fogel for helpful comments and suggestions. This work was partially funded by a University of Queensland Postdoctoral Fellowship.

References

Angeline, P.J. 1994. An alternate interpretation of the iterated prisoner's dilemma and the evolution of non-mutual co-operation, *Proc. 4th Artificial Life Conference*, 353–358.

Axelrod, R. 1984. *The evolution of co-operation*, Basic Books, New York.

Crites, R.H. & A.G. Barto, 1996. Improving elevator performance using reinforcement learning, *Advances in Neural Information Processing Systems* **8**, 1017–1023.

Epstein, S.L., 1994. Toward an Ideal Trainer, *Machine Learning* **15**(3), 251–277.

Fogel, D.B., 1993. Using Evolutionary Programming to Construct Neural Networks that are Capable of Playing Tic-Tac-Toe, *Proc. 1995 IEEE Int. Conf. on Neural Networks*, 875.

Hillis, W.D. 1992. Co-evolving parasites improve simulated evolution as an optimization procedure, in Langton et al., eds., *Artificial Life II*, Addison Wesley, 313–324.

Langley, P., 1995. Order Effects in Incremental Learning, in P. Reimann & H. Spada (eds), *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science.*

Michie, D. 1961. Trial and Error, in *Science Survey, part 2* (Penguin), 129–145.

Miller, G. & R. Cliff, 1994. Protean behavior in dynamic games: arguments for the co-evolution of pursuit-evasion tactics, *Proc. Simulation of Adaptive Behavior* **3**, 411-420.

Pollack, J.B. & A.D. Blair, 1998. Co-Evolution in the successful learning of backgammon strategy, *Machine Learning* (to appear).

Rosin, D. & R.K. Belew, 1997. New methods for competitive coevolution, *Evolutionary Computation* **5** (1), 1–29.

Rumelhart, D., G. Hinton & R. Williams, 1986. Learning representations by back-propagating errors, *Nature* **323**, 533–536.

Samuel, A.L., 1959. Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development* **3**(3), 210–229.

Schraudolph, N.N., P. Dayan & T.J. Sejnowski, 1994. Temporal difference learning of position evaluation in the game of go, *Adv. NIPS* **6**, 817–824.

Sims, K. 1995. Evolving 3d morphology and behavior by competition, Proceedings of the Fourth International Conference on Artificial Life, MIT Press, 28–39.

Sklar, E., A.D. Blair & J.B. Pollack, 1998. Co-evolutionary learning: machines and humans schooling together, *Workshop on Current Trends and Applications of Artificial Intelligence in Education*, Mexico (to appear).

Sutton, R. 1988. Learning to predict by the method of temporal differences, *Mach. Learn.* **3**, 9.

Tesauro, G. 1992. Practical Issues in temporal difference learning, *Machine Learning* **8**, 257.

Tesauro, G. 1995. Temporal difference learning and TD-Gammon, *Comm. ACM* **38**(3), 58.

Walker, S., R. Lister & T. Downs, 1994. Temporal difference, non-determinism, and noise: a case study on the 'othello' board game, *Int. Conf. Artificial Neural Networks*, 1428.

Zhang, W. & T.G. Dietterich, 1996. High-performance job-shop scheduling with a time-delay TD(lambda) network, *Adv. Neural Information Processing Systems* **8**, 1024–1030.