GTE Laboratories

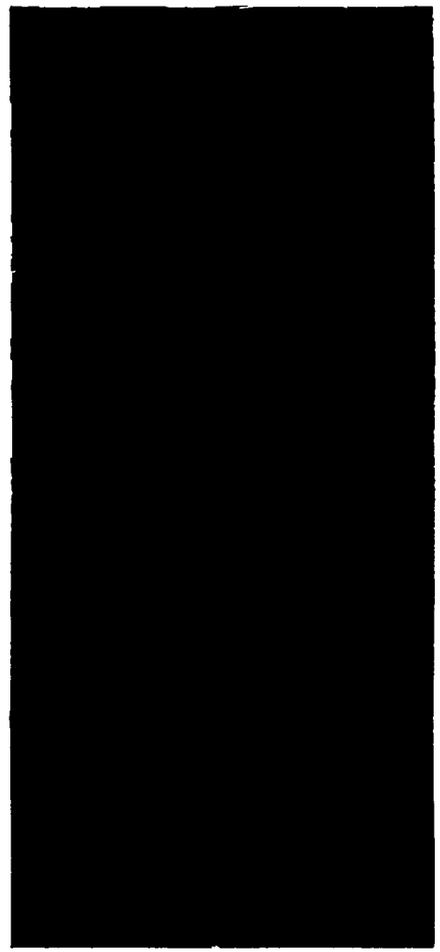
TR 0054-07-89-509

Adaptive Strategies of Learning: A Study of Two-Person Zero-Sum Competition

O. G. Selfridge

July 1989

Technical Report



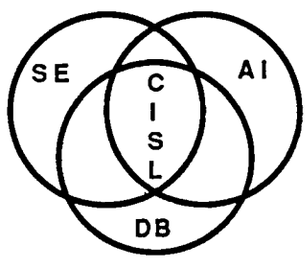
COMPANY PRIVATE
For Distribution Within GTE Only

TR 0054-07-89-509

Adaptive Strategies of Learning: A Study of Two-Person Zero-Sum Competition

O. G. Selfridge

July 1989



GTE LABORATORIES INCORPORATED
40 Sylvan Road
Waltham, MA 02254

TR-0054-07-89-509

This document is a GTE Laboratories Technical Report. It describes results and conclusions reached upon completion of a major phase of a research project. The ideas and views put forth by the author have been reviewed and accepted by the appropriate Laboratory Director.

Oliver G. Selfridge — Date 7/31/89

Manager — Date 7/31/89
Self-Improving Systems Department

Director — Date 7/31/89
Computer and Intelligent Systems Laboratory

Adaptive Strategies of Learning

A Study of Two-Person Zero-Sum Competition

Oliver G. Selfridge
GTE Laboratories Inc.

July 1989

ABSTRACT

We explore the use of adaptive components in strategies for playing extremely simple two-person zero-sum games. The adaptation is different from that usually considered in the field of Adaptive Control; rather it is a form of test and gradient descent. One of the more general conclusions that can be drawn from this work is that adaptive learning from experience seems to be far richer a topic and more complicated a process than would be imagined. There are many lessons here for studies in learning from experience in more complicated domains.

The general adaptive element here includes 1) an action cycle, which exercises some control; 2) a testing cycle, which informs the strategy, through 3) its evaluation function, which way and how much to alter the parameter of its strategy. Such an element may itself be controlled with adaptive loops, and those hierarchical systems can exhibit surprisingly subtle and powerful behavior. This paper explores the use of such adaptive procedures on variables that can attain some range of real values. For variables restricted to integer values, say, the procedure must be much modified. That is, what is needed in the general case is a variety of adaptive elements, not just one.

In the experiments done here, it turns out that convergence to *minimax* is the exception, and behavior in most cases tends to some kind of a joint limit cycle. The general result is that such competitive systems usually do not converge to any kind of stability. An interesting application of this is to distributed or hierarchical control systems where different control units may be considered to be competing for some resource.

Acknowledgments

I am greatly indebted to my colleagues at GTE Laboratories, especially John Vittal, Rich Sutton, and Judy Franklin, for their support and continuing constructive comments. A highly abbreviated form of this paper was presented under the same name at the Sixth International Workshop on Machine Learning in June at Cornell in Ithaca, New York; and published in the Proceedings [OGS89A].

CONTENTS

FIGURES

Adaptive Strategies of Learning

A Study of Two-Person Zero-Sum Competition

Oliver G. Selfridge
GTE Laboratories Inc.

1.0 Introduction

1.1 Adaptive strategies in two-person zero-sum games

The underlying purpose of this study is not to study games, simple or otherwise, but rather to begin an exploration of certain kinds of distributed or hierarchical control systems where different control units may be considered to be competing for some resource.

1.2 Purpose

It will turn out that merely making strategies subject to modification and control adds a surprising richness to the possibilities of the strategies themselves, even if they are extremely simple. The very concept of strategy can be enlarged to include the changes to be made to a strategy as a result of the winnings or losses made from it. The purpose here is to explore some of those implications, and to illustrate them with simulations using an exceedingly simple two-person game.

1.3 Definitions

To define terms first: two-person games are played by two *players*, each playing to win. Winning may be binary, as in *ducks and drakes* or *tennis*, or it may have a reward with any of many values, as in *poker*. If the reward for one player is the negative of that for the other, then the game is said to be *zero-sum*. Many popular games are zero-sum, like chess, checkers, and poker; and including baseball, scrabble, and tennis. For the purposes of this paper, a game is a *competitive interaction* that rewards its participants with something like money. We are not concerned here with the other kinds of rewards, like the enjoyment of playing by itself. Non-zero-sum games are those where the total value of wealth changes as a result of the interaction: examples include collaborating on looking for buried treasure, dancing, and so on. Many zero-sum games are *symmetrical*, with both players acting on an equal footing, save perhaps that one player moves first. These are the kind of game we are concerned with here.

1.4 Players and strategies

The participants or players play according to *strategies*: by that I mean that the actions they elect to take—the *moves*—can be described in some way so as to encompass their behavior. Sometimes the description of a strategy can be made easily, as in the game we use here, and sometimes it is very difficult, as in *tennis*. A player exhibits a strategy, therefore, to the extent that he adopts the same behavior in the same situations. Note that one should not say "makes the same move," because a strategy may well use a *random component* to select the actual move—we shall do this in the strategies discussed here.

1.5 Strategies and learning

Our concerns are with two-person zero-sum games, the strategies that are used to play them, and how those strategies can be modified so as to play better. The particular kinds of modifications are those derived from experience of playing. This places their study in the realm of learning by self-improvement, to distinguish them from learning from instruction and other kinds of learning. On the whole, learning is learning from experience: if we look at all the instances of behavior that we call learning in living beings, the vast majority of them are learning from experience. Furthermore, both in people and in animals, learning from experience obviously precedes learning by instruction—children learn to talk before they can be instructed. And much of our knowledge has been implanted genetically, so that one must consider evolutionary learning.

In Artificial Intelligence (*AI*), the field of Machine Learning—that is, learning by the computer—has a special role; truly intelligent software, like people, must be able to adapt to changing circumstances, changing environments, and changing tasking.

There is clearly a vast range of adaptive changes that can be termed learning; and the range of organisms that can exhibit and take advantage of adaptive or learning behavior is similarly vast. It is desirable that the whole gamut of learning be examined, even if it is not modeled. This paper is concerned, obviously, with some of the most elementary and primitive—"atomic"—mechanisms of adaptive improvement that underlie all kinds of behavior that can be called learning.

It is worth stressing that the particular game that is the environment here is unimportant, and it provides merely a vehicle for changes in strategies and for evaluating those changes.

2.0 Adaptation and Learning: Generalities

In Artificial Intelligence, adaptation and learning have some overlap, but the consensus is that there is not much. A servomechanism or any similar feedback control is not intelligent, though it may be necessary for intelligence. Most of the work in Machine Learning is concerned with *symbolic learning*, that is, with learning that involves symbols, like that which goes on in schools or universities. One of the arguments here is that simple adaptation can, if applied generously enough, eventually perform what looks like symbolic learning.

2.1 Strategies and their trajectories

The player then, if he is to be said to be learning from experience, must modify the strategy used to play the game, and do so on the basis of how well it has been working. One might imagine the successive strategies as forming some kind of trajectory through the space of strategies defined by their describable attributes. Such trajectories will provide appropriate insights only in some cases (one of which is explored here), including especially when the strategies are described by variables that have continuous effects: one might imagine that certain kinds of economic behavior are such cases. Indeed, the game considered here is perhaps the simplest possible environment for a variety of economic considerations that apply to questions of competitive behavior. These questions and their implications are considered in more detail in Section 10.

In some sense, a strategy that can be changed may be just a constant strategy, if the ways of changing the strategy are tightly enough specified. For example, suppose in some game that a player plays in one way when he is winning, and another when losing. Each *substrategy* can be regarded as a strategy in its own right; but the two together, with the rules for switching between them, also form just one strategy.

A key question will be the representation of a strategy: for example, should every turn be spelled out?—or should merely the general rules for determining or selecting the turns? It is clear that the means of expressing or representing a strategy is important in delimiting the strategies that are usable. The strategies here may be described with merely a set of numbers.

2.2 The principles of learning from experience

What one learns from experience obviously depends crucially on how that experience is represented.

- If the experience is represented in terms of specific observations and specific actions taken in response to them, then there is the hard question of how to make inductions and generalizations from that history.
- If the experience is represented more generally, the question still remains of where the generalizations came from? Were they learnt, or were they hard-wired?
- How much of the detailed experience stored in the history is irrelevant?

These questions and more are the stuff of representation, always a trenchant point in AI and machine learning. There are other simple observations and principles about learning from experience:

- Try what has worked before. If that doesn't work, try something else.
- Faced with a problem or decision, try what has worked in similar situations.
- If it ain't broke, don't fix it. [folk knowledge]
- If at first you don't succeed, try again.

2.3 The learning model

Since we are concerned with learning, rather than with teaching or instruction, we use a simple model of the learning process, which we term *self-improvement*; in its simplest form, it may express mere adaptation or tuning. Basically, the system or organism undertakes some action, observes what happens, evaluates whether that was good or bad, and corrects or extends the action on the basis of that evaluation. Almost every term in that sentence may need further explanation:

- "undertakes some action": this means e*xerts some control;* that is, sets or resets some control parameter, like a switch or a setting. The kind of control exerted will greatly depend on the control system itself and its environment and how they all work. That control will have some effect on the environment, which will respond with some changes in its *observables*. The action taken will form part of the execution of a strategy.
- "observes what happens": this means, typically, sees how good the performance is. In many situations, other data may be noticed that may or may not be useful in improving the strategy.
- "evaluates ...": this means analyzes the date and evaluates it—is it better or worse than the previous performance or payoff?
- "corrects or extends ...": this again means e*xerts some control;* but the actions taken may be different from the testing in the first step. It will depend on the system design and the representations.

2.4 Earlier work; some history

The game that is of concern here is Penny-Matching (*PM*); it and its variants have been widely known for a long time. In the early 50s, both Dave Hagelbarger and Claude Shannon at the Bell Telephone Laboratories built relay machines that would adaptively predict binary choices made by people, using their previous choices [HAG56].

Since then there has been very little attention paid to *PM*, which seems to me to be perhaps the simplest two-person game of all. Jeff Barnett discusses the ordering of trials of methods to achieve a simple goal, in this model:

> "... to satisfy a goal ... There is a tradeoff between resources spent executing a method and those spent selecting the method since both method selection and execution are part of the problem-solving activity." [BAR84]

But that case is not really parallel to the one discussed here, because the underlying concerns are different: Barnett is interested in the value of *Control Knowledge* in helping to order the trials, while the concern here is more with the evolution of strategies.

In more recent work [WIN88a,WIN88b], Windecker at Bell Labs discusses adaptive automata that can do better against a nonrandom opponent:

- "Game theory says that the strategy that minimizes a player's maximum loss is to play heads [*H*] and tails [*T*] with equal probability, 0.5. Here we describe how a nondeterministic ...

> adaptive automata [*sic*] ... can learn over a sequence of plays, to take advantage of nonrandom patterns of play of its opponent ... " [WIN88b, p. 1]

Although there is a substantial overlap in the interests of those two papers and this one, there are substantial differences too. The implementation details—the units of the networks—and the nature of the opponent are treated very differently. This paper is not concerned with implementation at all; and in the Windecker work, "... we make the ... assumption that B [the second player] does not play randomly and is incapable of changing his nonrandom pattern ... " His adaptive automata play against nonadapting opponents. These may be of two kinds, deterministic and nondeterministic; in either case the fixed strategies include short term consistencies, like repeating a particular series of three moves, and the adapting strategy can detect and take advantage of them.

3.0 The Domain and the Givens

3.1 Penny-matching

The domain of concern here is what must be the simplest possible two-person zero-sum game, *penny-matching*. It is well-known: each player privately chooses one of *heads* or *tails*: the first player wins if the choices are revealed to have been the same, the second if different.[1] In [VonN48], it is used as an example to illustrate the minimax principle; and it shows that choosing heads and tails randomly with equal probabilities will ensure that on the average the gain or loss is zero. Of course, so is the profit. It is perhaps superfluous to add that the concern here is with strategies expressed over considerable periods of time, playing very large numbers of individual games or choices.

3.2 Strategies and their representations

We are not concerned here with the details of penny-matching. We limit the strategies to those that merely specify a *Probability of Saying Heads* denoted here by *P*; beyond the most elementary cases, that Probability will change.

3.3 Assumptions of ignorance

It is also assumed that the players are not like the readers, who know the nature of the game and its payoff function, and who may be able to analyze it. In fact, it is assumed that the only knowledge a player has about the game is what happens when he plays it, that is, what *P* he selected, and what the payoff was as a result. The game should be thought of as played in this way: Each player has in front of him a dial that he can set, and a counter that tells him, immediately after each move, how much he has won or lost. The dial runs from 0 to 1, and represents the value of *P*, the probability of selecting *heads*. Neither player needs to be directly aware of the other, or of the other's actions or outcomes; neither needs to know that this is a two-person zero-sum game.

---

[1] A moment's thought will reveal that this game is isomorphic to others like *Scissors, rock, and paper*, if one ignores ties.

This assumption that each player is using a *P* strategy may seem like a very great restriction. So it is, but the concern here is not with the game itself, but with how adaptation and learning can take place while it is being played.

3.4 The players' purposes

We say that each player's purpose is to maximize his profit. However, that does not by itself provide a tight definition. The first player sits watching his outcome: how does he know whether it is good or bad—not at an absolute level—but with respect to what he *might* get? That is, we suppose that the player does not even have the knowledge that he is playing a zero-sum game. That means that, whatever the rewards happen to be in some local duration, the player will always be experimenting with his strategy to see if he can do better.

If his opponent, as was suggested before, is constantly playing heads, then the first player ought to be playing heads as well. Even if his opponent is playing heads only slightly more frequently than tails, the first player ought to play heads all the time. So how should the players determine their success or failure? They can inspect only the outcomes, and what determines those will be changing. Over how long a period should each integrate his outcome to determine how good a recent strategy change was? The analogy for business games is the assessment of a firm's discount rate in making business projections, in balancing short term profit against longer term profit.

3.5 *PM* and its payoff function

The players are ONE and TWO: let us suppose that they play *PM* for a large number of games, selecting heads at every move with some fixed probabilities P_1 and P_2. For each game, the result for a player is given by the *payoff matrix*, which is a function of what each player did, say T_1 and T_2, both provided from the *P*s by some random number generator; see Figure 3.1.

		Player 2 says: Heads	Player 2 says: Tails
Player 1 says:	Heads	-1.0	1.0
	Tails	1.0	-1.0

Figure 3.1: Payoff matrix for single game of *PM*

As the number of games approaches infinity, and if neither player changes his strategy, then the *average payoff* can be computed from elementary probability theory:

$$\text{Average Payoff} = (2P_1 - 1)\,(2P_2 - 1) \qquad (1)$$

That function is shown in Figure 3.2. The joint *minimax* solution[2] is in the middle of the plot, at the point (0.5,0.5), but the minimax payoff is attained when either $P_1 = 0.5$ or $P_2 = 0.5$. In the figure, the two shaded quadrants show where the game is unprofitable for the first player. The overall topography of the function is typical, a saddle point or *col*. Much of the analytical difficulty arises from the fact that the strategies include a random element, so that the payoff function is reached exactly only after an infinite number of plays. But here we are interested not in the analytical aspects, but rather in the heuristic and experimental ones.

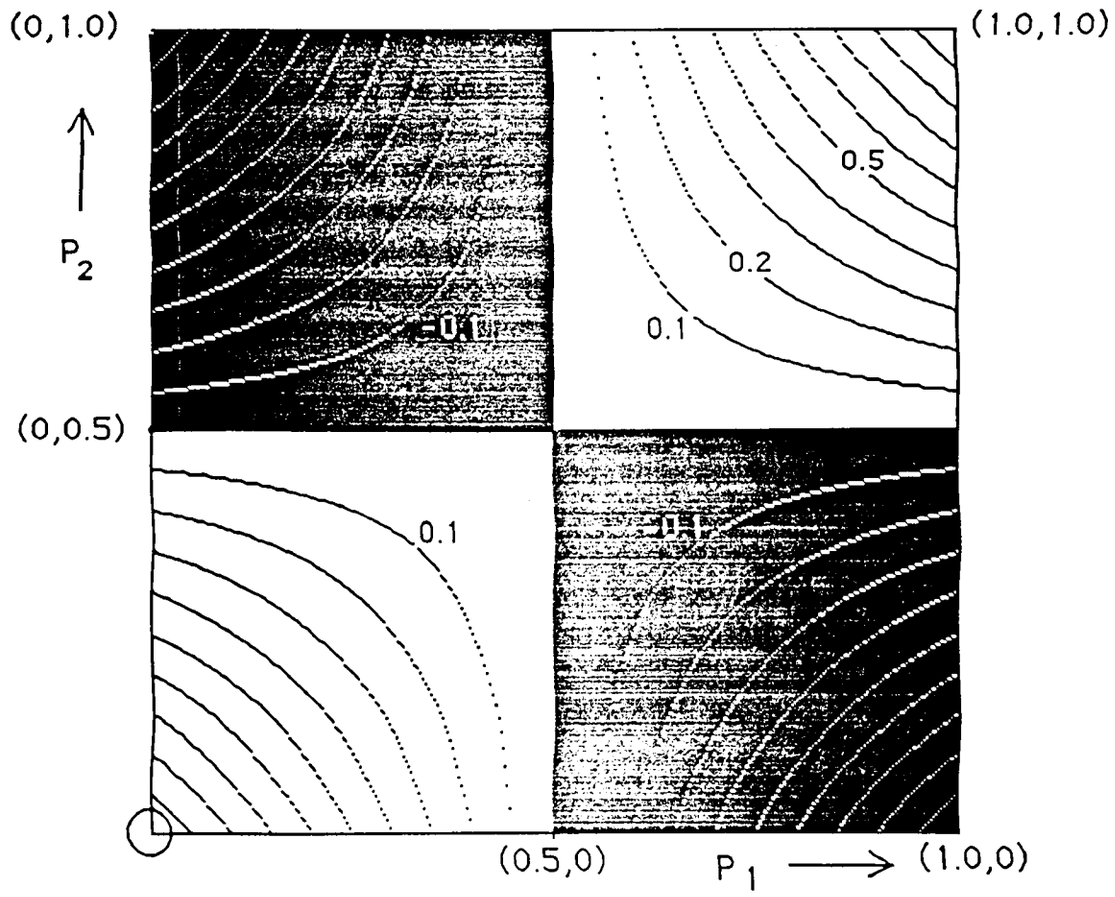


Figure 3.2 Payoff average for fixed probability strategies

---

[2] For zero-sum games, the player's choice is to select a row in the payoff matrix; since his opponent is selecting a column, the player can assure himself of losing the least by selecting a row that has the least reward for the opponent—that is, to select the row with the minimum maximum, or minimax. The term is due to Von Neumann [VonN48].

First we summarize certain conventions:

- The two players are ONE and TWO. Each game is two simultaneous *moves* made by the two players. A move means to specify a binary choice, heads or tails, represented respectively by 0 and 1. Games occur in a numbered sequence. If both players make the same move, then ONE wins; if not, then TWO.
- The k^{th} move by player ONE is $T_1(k)$; by TWO, $T_2(k)$. ONE wins the k^{th} game if both players choose the same, that is, if $T_1(k) = T_2(k)$; TWO otherwise.
- For the k^{th} game, ONE and TWO have winnings (negative winnings are losses) $W_1(k)$ and $W_2(k)$ respectively. Since *PM* is zero-sum, $W_1(k) = -W_2(k)$. For *PM*, $W_i = \pm 1$. Cumulative winnings, or the players' Fortunes, through the k^{th} game, are

$$F_i(k) = \sum_{j=1}^{k} W_i(j) \qquad (2)$$

- Each player uses a *P*-strategy. That means: at each game, each player inspects his P_i, and then inspects the output of a random number generator that outputs a number between 0 and 1; when that output is less than P_i the player chooses 0 or heads for his turn; otherwise, 1 or tails. The P_i will in general not be constant.
- For any variable X, $\overline{X}$ is its average or expected value.

4.0 Strategies and Learning

In this paper, experimental results are reported about a range of adaptive strategies. They will illustrate the development of complex strategies as incremental improvements in simple ones.

Minimax is the appropriate strategy for a player for whom the likelihood of any loss must be minimized; that is, for whom the objective function is solely to minimize the chance and size of a loss. The concept of minimax derives from the *payoff matrix*, shown in Figure 3.1. In classical analyses of game theory achieving minimax is used as the primary goal for the players.

For a single game of *PM*, consisting of a move by both players, there is no minimax solution—either way it is possible for a player to lose. Now consider a long or infinite sequence of games, in which the players have picked heads with probabilities P_1 and P_2. Then we need, not a payoff matrix, but a payoff function, which was given by the RHS of equation (1), as plotted in Figure 3.1. Now for this game, there is a minimax solution, and it is that for either player $P_i = 0.5$.

It is commonly assumed that a minimax solution is the *best* solution; but that is true only if best means avoiding losses. Supposing that TWO is stubbornly choosing heads every time: it is clear that there is a chance for some positive profit for ONE, by merely agreeing with him. It can be argued that TWO may thereby be trying to "trap" ONE into changing his strategy. Although that argument does not hold water—I do not perform the analysis here—the point is irrelevant to the fact that sometimes it is possible to adopt a non-minimax strategy that wins. That is, to the extent

that either player is not adopting the minimax strategy, then profit is possible (and so is loss) for both players. This study is concerned with that kind of competition between them.

The primary learning tool used by the players in this model is elementary: try different strategies—only slightly different—and then move your basic strategy the way that seems to work out better. This corresponds exactly to an extremely simple adaptive method called elsewhere an *atom* of learning [OGS89B].

The overall architecture of adaptation depends crucially on the representation of strategies and modifications of them. What kinds of strategies can be tried and adopted? We are limiting our players to *P*-strategies; that is, to strategies that basically select a *P* and then make small changes to it. There is of course an enormous number of other kinds of ways of making and representing strategies:

- A player's move can be a binary function of the previous moves made by either or both players. There are many easy ways to represent such a strategy. Of course, if the details of the strategy are known to the opponent, he can very likely devise his own to defeat it.
- The sequence of moves by a player can be represented as a binary number: perhaps the oddness or evenness of the decimal digits of π. Note that this strategy may look like a *P*-Strategy with $P = 0.5$; in fact, the similarity goes beyond that because the random number generators on computers are not truly random at all.
- It will no doubt have occurred to the reader that if he finds himself losing while playing, all he has to do to is to change his *P* to 1-*P*. There are reasons why that tactic was not incorporated into the strategies discussed here; first, it is a complication of what ought to be something very simple; and second, to adopt that tactic supposes that the player has an extra piece of knowledge, that *the game is zero-sum*. Nevertheless, it might be instructive to perform some of the analyses with that tactic added to the strategies.

In all those cases, it is clear that a fixed strategy tends to have some degree of vulnerability. In that sense, then, a strategy should be adaptable; that is, its local temporary tactics should be changeable according to circumstances.

But first one should look at the precursors in a little more detail.

5.0 *FP* Strategies

The competition between two fixed-*P* strategies (*FP* strategies) is straightforward: in the long run, the payoff is approximately what was shown in equation (1) and Figure 3.1. That is, for *N* games, the expected cumulative winnings are

$$\overline{F} = N\,(2P_1 - 1)\,(2P_2 - 1) \qquad (3)$$

and, of course, the runs will be different, because of the randomness of the individual moves.

What, then, is the best *FP* strategy against another one? Examination of equation (3) shows that the best P_1 against an $FP(P_2)$ is

$$P_1' = \begin{cases} 1.0 & \text{if } P_2 > 0.5; \\ 0.0 & \text{if } P_2 > 0.5; \\ \textit{anything} & \text{if } P_2 = 0.5 \end{cases} \quad (4)$$

(remembering that $0 \leq P \leq 1$), and that the corresponding cumulative payoffs tend to

$$\overline{CW} = N\,|(2P_2\text{-}1)|$$

This is shown in Figure 5.1.

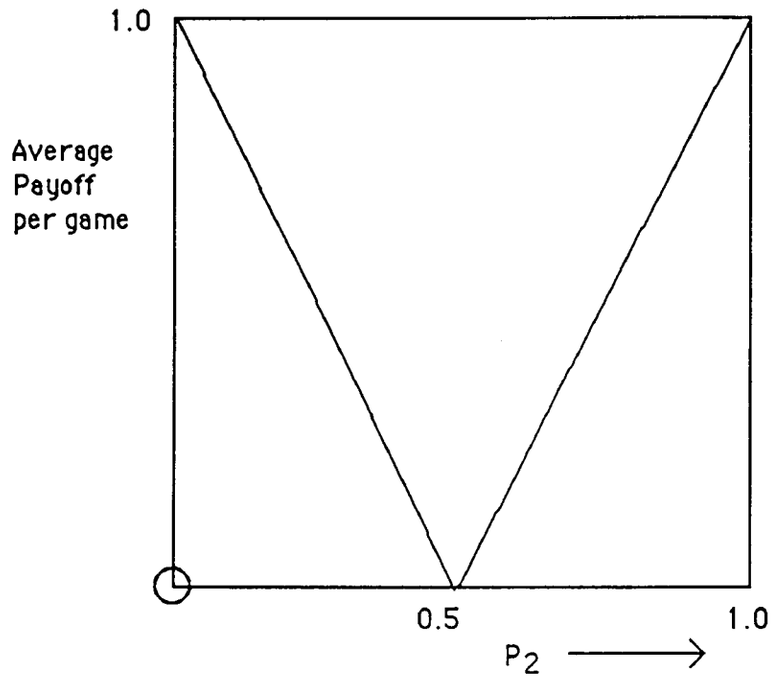


Figure 5.1. Payoff using best *FP* strategy against *FP* strategy with $P = P_2$.

It must be admitted that such games are dull indeed, although the number of people who play the lottery (an even duller game) would suggest that some do not think so.

6.0 Adaptive P-Strategies or *AP* Strategies

The basis of the adaptive strategy is to make an estimate of whether it pays to increase or decrease P by trial; and, if so, then to make the appropriate change. In essence, the strategy finds out some kind of reward or evaluation gradient in the strategy space (albeit one-dimensional), and essays some steepest ascent. This technique is well known.

Here is one expression of such a strategy:

- A square wave is added to $P(t)$ with some period *PER*. The point is that some changes must be tried; although there is nothing inherent in the approach that requires those changes to form a square wave, that makes the computations more straightforward and easier to program. Its period or frequency may seem to be unimportant, providing that proper records of the different accumulated payoffs are kept; but some subtleties will be seen to arise.
- The added square wave has an amplitude *AMP*. In the context of the problem *AMP* ought to be small compared with 1, because one of the pieces of knowledge hard coded into the problem is that probabilities are constrained to lie on the interval (0,1).
- The added square wave also has a phase *PH*. Like the period, this may seem to be irrelevant, but in fact it will play an important part in the later elaborations of adaptation.
- The evaluation function is computed by comparing the two accumulated payoffs during the trial period. Their difference is used to compute the change that is added to the probability *P*.
- The difference is multiplied by a Gain *G*, to provide the actual numerical change in *P*. As in the usual form of control theory models, a gain that is too large will lead to behavior that is rather like instability; one that is too small will adapt very slowly.
- The program has to detect recent changes in payoffs, especially those that derive from recent changes in *P*. That means that it should give recent events more weight than those less recent. An extreme procedure would be to store anew the two payoffs to be compared after each change in *P*, which is what is done here in the next section. Alternatively, one could continually add small changes to *P* as a result of some exponential weighting of recent payoffs; such a procedure can be computationally very simple.

All this seems like a very general type of strategy; and indeed it does handle a broad class of problems. But there are some real limitations in what such a strategy can handle, which will be discussed in Section 10.

The strategy can be specified in detail:

- Add a $2\Delta P$ square wave of some period to P:

  $$P' := P + \Delta P \qquad \text{and} \qquad P' := P - \Delta P \tag{6}$$

  P' is the probability that is actually used to play with.

- Add the winnings separately for the times when P is increased and decreased; then take the difference:

  $$\Delta F = F(P + \Delta P) - F(P - \Delta P) \tag{7}$$

  where from equation (2) $F_i = \sum_{j=1}^{k} W_i(j)$.

- Modify the basic P by adding the difference in those two accumulated winnings multiplied by some G.

$$P := P + G\,\Delta F \tag{8}$$

subject to the usual constraints that $0 \leq P \leq 1$. Typically, G is termed the *Gain* of the control loop.

- And keep on repeating this process.

The reasoning behind the model strategy is purely analogue; in section 10, we shall examine some modifications: for example, section 10.3 considers the use of sine waves instead of square waves; section 10.6, the use of a nonlinear function in place of the RHS of equation 8.

7.0 A Program for Exploring *AP* Strategies

The ideas discussed above were implemented in a small program on a Mac II, described in the Appendix.

It is important to verify first that the adaptive strategies do in fact provide approaches to the optimum values specified by equation (4) and Figure 5.1. Now the speed of approach depends crucially on the particular value P_2 of the *FP* strategy. As is clear from Figure 5.1, the clues about which way to move are smaller the closer P_2 is to 0.5. Figure 7.1 shows a number of traces or time plots that verify the effectiveness of *AP* strategies against *FP* ones. Note that the adaptive strategy against a random (P=0.5) strategy looks like a mere random walk, as required by theory.

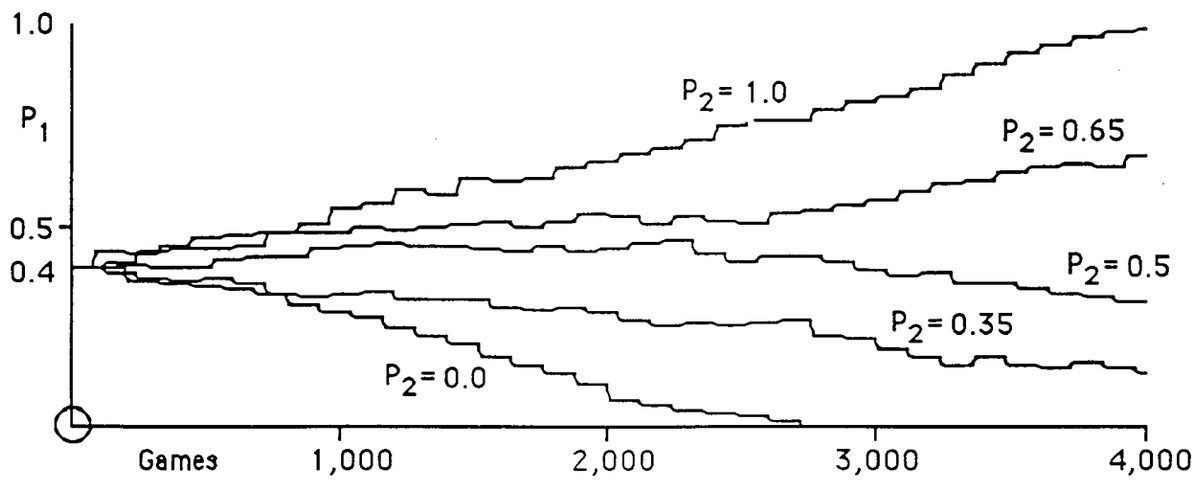


Figure 7.1. An *AP* strategy adapting against different *FP* strategies

- The *FP* strategy was P_2, and the *AP*,was P_1, starting at a value of 0.4. The period was 120, and the gain 0.001. The individual games were simulated with a standard random number generator, whose effects can be seen in the irregularities of the tracks. If the program is run for much longer times, the tracks convincingly converge to 0 or 1, except for P_2 values close to 0.5; just as theory says.
- This and subsequent figures have their parameters listed in the Appendix.

Those plots were made from the Mac II program by merely specifying the parameters of one of the *AP* strategies in the appropriate way.

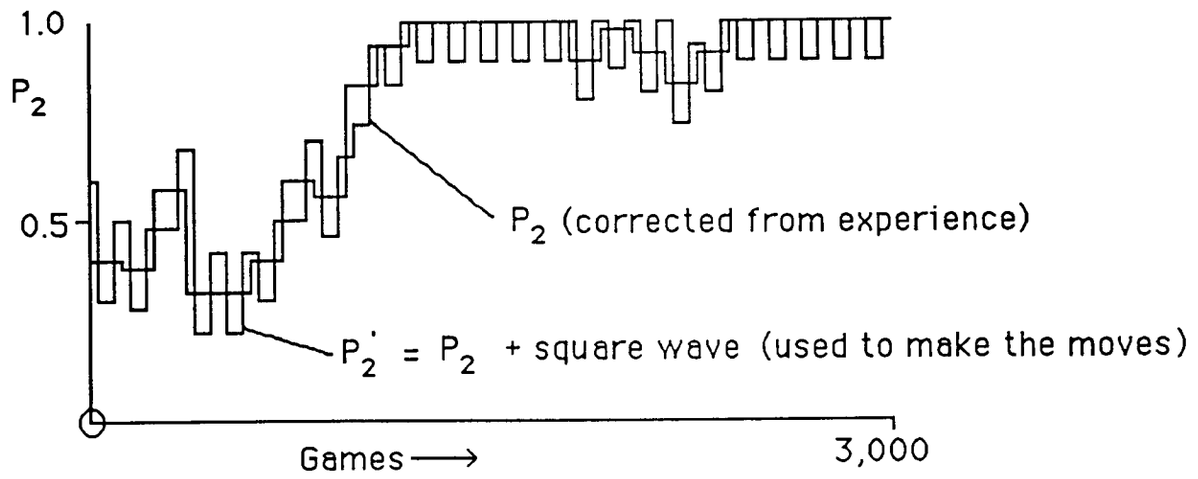


Figure 7.2: An *AP* strategy adapting P_2 against a (P_1=0.75). P_2 starts at 0.4, and grows irregularly to the optimum, 1.0. (Parameters in the Appendix).

The detailed history of an adapting *P* is made clearer in Figure 7.2, which shows both P_2 and P_2', described in equation (6); remember that P_2' is the value actually use to play with. That is, the latter moves up and down on a regular basis, testing whether to see which way to change. Here TWO is playing against a fixed strategy (*FP*) by ONE, and $P_1 = 0.75$.

The robustness of the adapting procedures is one that obtains with many hill-climbing techniques.

7.1 Trajectories and history plots

In general, in the figures below, the added square wave is assumed, and what is plotted is merely the basic *P*s. For the competitive cases that are the subject of this paper, there are two kinds of figures: tracks or time plots, as in Figure 7.1; and trajectories as in Figure 7.3. In trajectories, the time, or number of games played, is a hidden variable. Compare that figure with Figure 7.4, which shows the corresponding time histories for the same data.

The trajectory in Figure 7.3, which covers the first 60,000 games, is locally extremely irregular, as a result primarily of the random number generator (RNG). Overall, it is a wide and loose expanding loop, which in fact eventually "bumps into the stops" that are the limits 0 and 1 for a probability. That is, for this particular choice of strategies, with its parameters, the behavior of the two players settles into a limit cycle, or, rather, a very broad "limit annulus;" which results from the detailed interaction among the individual local behaviors and from the effects of the RNG.

It is perhaps desirable to separate the general system behavior from that caused by the RNG of the implementation; and so another case will be considered in detail. Instead of using the game whose output is a binary number generated by a RNG, we can consider the game whose output is exactly

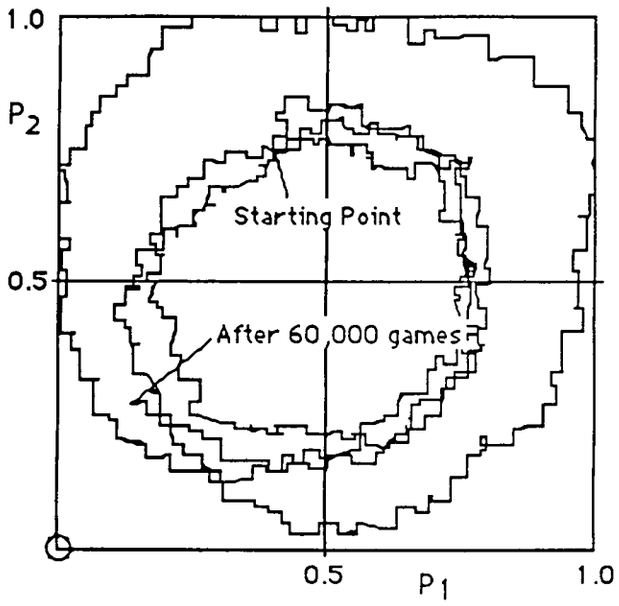


Figure 7.3. Trajectory of P_1 and P_2 both adapting for 60,000 games.

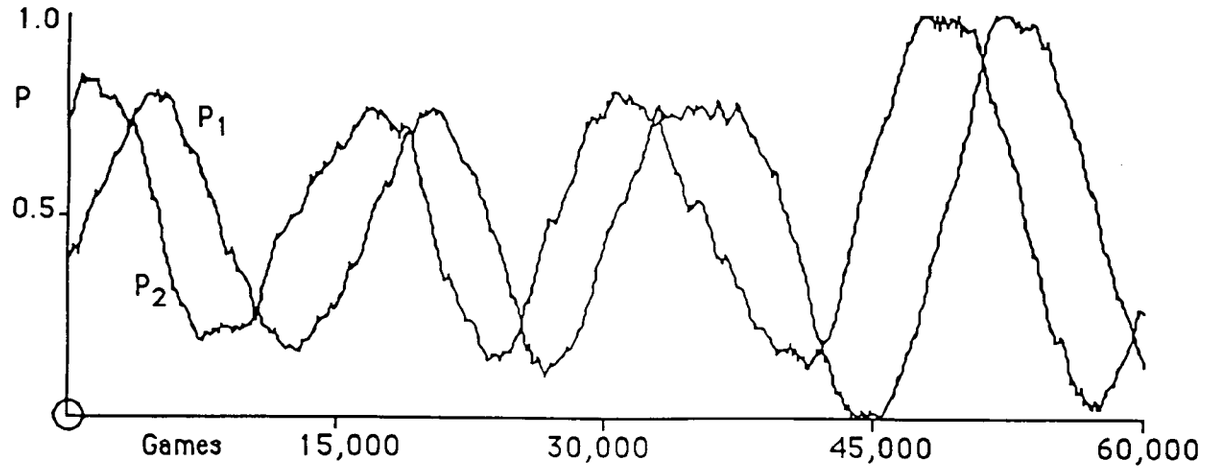


Figure 7.4. History of P_1 and P_2 adapting for 60,000 games.

that given by equation 1; so that each game may be considered as the averaged result from playing a very large number of games of the binary kind.

If we do run the program with that average payoff function and the same parameters as in Figures 7.3 and 7.4, we get Figures 7.5 and 7.6. The trajectories and histories are of course a great deal more even.

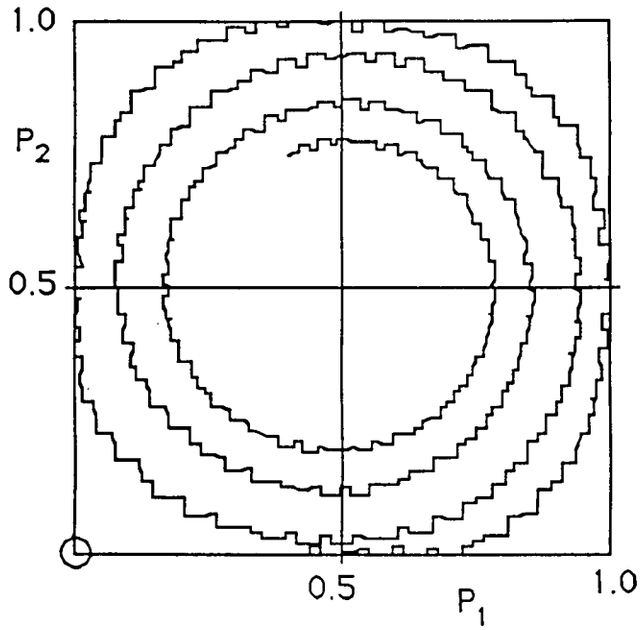


Figure 7.5. Trajectory of P_1 and P_2, as in Figure 7.3, but using the Payoff Function directly; 60,000 game equivalents were played.

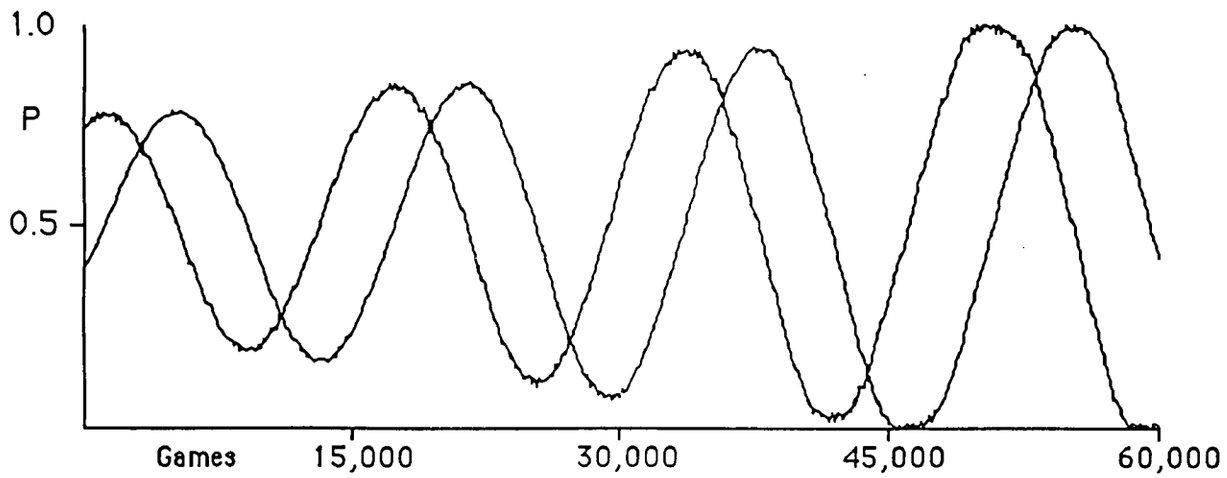


Figure 7.6. History of P_1 and P_2, as in Figure 7.4, but using the Payoff Function directly; 60,000 game equivalents were played.

7.2 Trajectories go clockwise

A key feature of the examples in the figures is that the trajectories do not usually seem to converge to any kind of minimax; but there are exceptions, for example, Figure 8.1. Instead, there seems to be a kind of loose limit cycle, which is converged to both from inside and outside; this is shown in Figure 7.7. The shape is circular or elliptical, depending on the gains and other parameters of the individual control loops. As the next section will discuss, the limit cycles are usually bounded by the stops which represent the limiting values of 0 and 1 for probabilities.

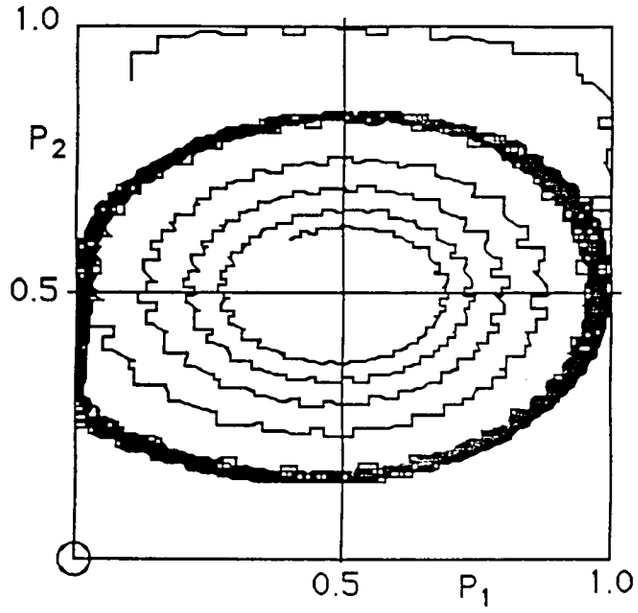


Figure 7.7: Convergence to a limit cycle, from inside and out. The shape is not circular, because the two controls had different gains.

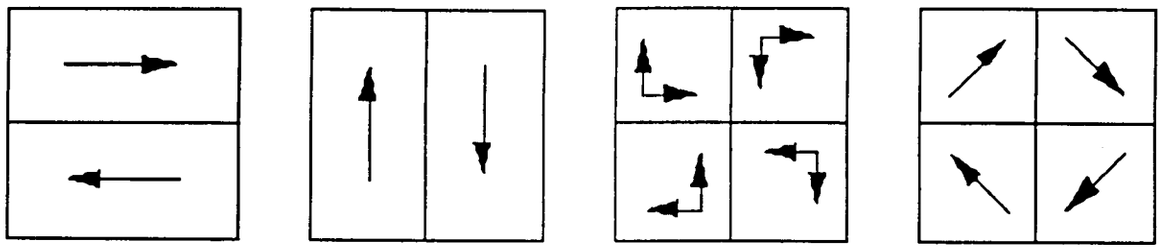


Figure 7.8 Why trajectories go clockwise.

The course of the trajectory is clockwise, which results merely from the pictorial representation and the assignment of *matching* to ONE. Quite generally, the trajectories go clockwise. The reason for this is easily understood from Figure 7.8. Figures 7.8A and 7.8B shows the tendencies of P_1 and P_2 to drift in the corresponding half-squares; Figures 7.8C and 7.8D combine the two and their drifts.

7.3 Measuring the gradient

The behavior of the joint system depends on surprisingly small differences in detail. In the basic testing sequence, the player alternately adds and subtracts some ΔP to his P. The strategy first adds ΔP and computes the payoff with that P'; then it subtracts ΔP and computes the payoff again. The difference, as in equation (7) is then multiplied by the gain and used to correct or adapt P. Now if P is too large, the idea behind adaptation is that the second payoff computed (for the smaller P') will be larger than the first. But if the payoff is falling rapidly, because of the

correcting *by the other player*, or *for any other reason*, then the payoff in the second half of the cycle will be correspondingly reduced. If the payoff is rising, the effect is of course reversed. In both cases, the strategy may move away from the local optimum. This phenomenon is illustrated in Figure 7.9.

The implemented program took advantage of that. Since the basic period was divided into four parts by that procedure, that is the value that is set by the user—in effect, the user sets a quarter-period, and that is what is shown on the displays.

This bias occurs because the injection of the modification occurs at a time when the first moment of the added square wave is maximally positive. The answer to that is to make the testing cycle—the square wave—uncorrelated with any trends at least; one way to do that is to use the basic cycle shown in Figure 7.10, where the adaptive corrections are added a quarter of the way through the cycle.

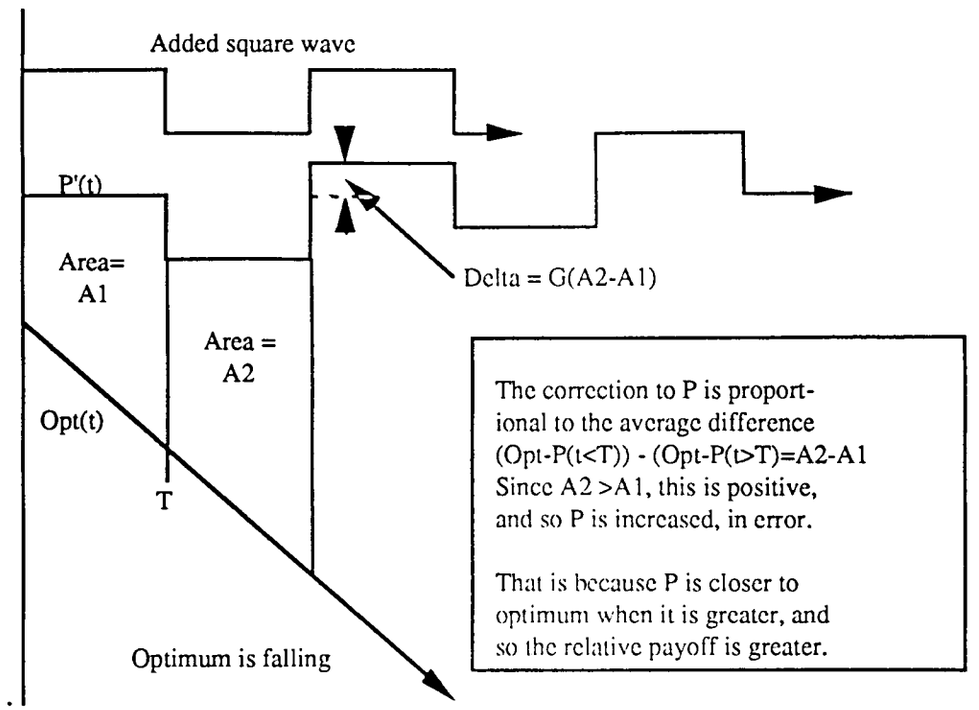


Figure 7.9 Some kinds of adaptation work badly in changing environments.

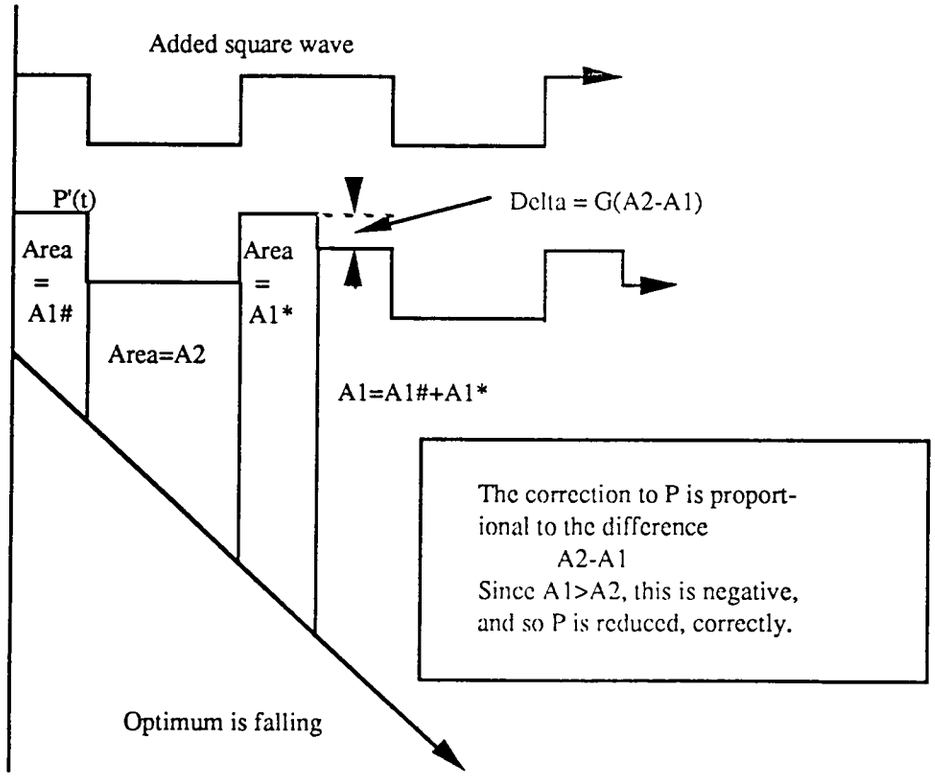


Figure 7.10 A better way to measure a 1-D gradient; it will be unaffected by linear changes in the environment.

8.0: Some experiments

This section explores the behavior of the competing system for various values of the parameters, showing the effects of the individual parameters

8.1 The parameters of control

There are several parameters that govern the makeup and behavior of each control mechanism, some of which are implicit:

- The nature of the added function to the variable being controlled. I have implicitly assumed that this function is to be *added*, but that is not an essential part of the notion. Section 10.6 shows an example of a *multiplicative* testing function used on the data that generated Figures 7.5 and 7.6.
- The *shape* of the added function; most of the figures in this study used a square wave, but others work as well—Section 10.3 shows examples of a sine wave and a random function.
- The *amplitude* of the added function.
- The *period* of the added function; and, in some cases, its *phase* as well. The period is in the nature of an integrative time constant.

- The nature of the corrections to the variable being controlled, which for most of the figures here is additive; as with the nature of the added function, that is not essential, and Section 10.6 shows a multiplicative correction. Furthermore, the systems here apply corrections once every cycle; that is not essential either—Section 10.5 also shows the case of making corrections continually.
- The nature of the evaluation function. It is usually assumed that the purpose of the system is to maximize the expected value of the payoff, in some sense of long run; of course, that is not essential to the model. In many realistic cases, for example, there is an assymmetry in the utility of positive and negative rewards; consider a corporation that might make many millions of dollars, but if it initially loses a hundred thousand it will fall into bankruptcy.

8.2 The effects of the parameters

Note that there are sets of parameters for which the system converges to something like minimax; an example is shown in the left side of Figure 8.1. But the usual case is the limit cycle, as in the right side of the same figure; this also emphasizes the importance of small changes in the parameters, because the only difference was that the periods of the two adapting loops were exchanged. In the figure, the starting point was (0.8,0.8), and the other parameters, as usual, are given in the Appendix.

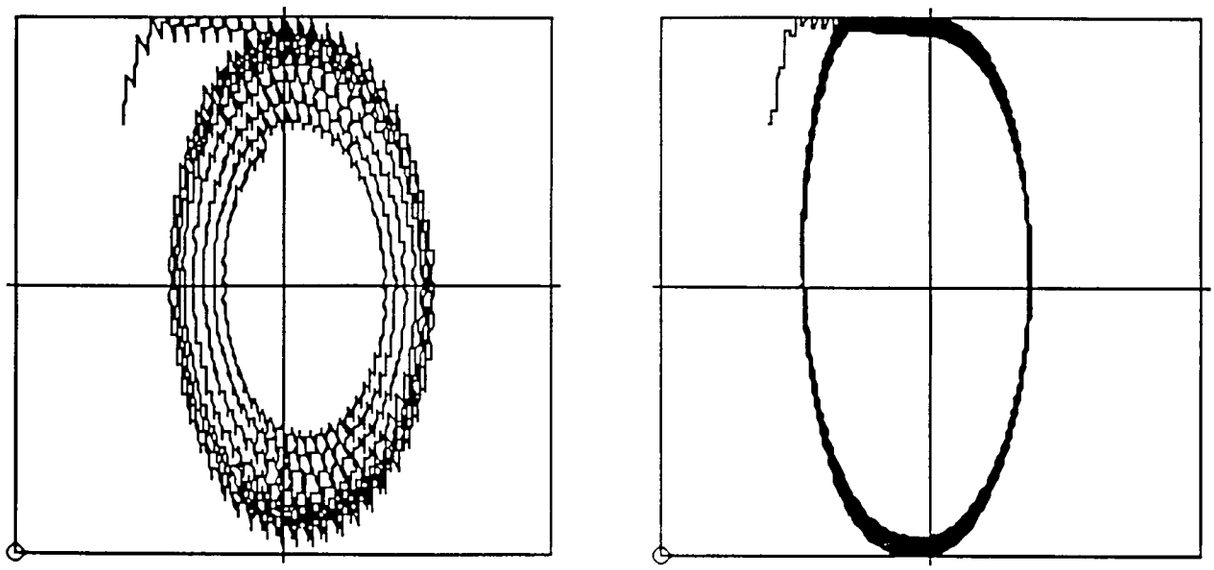


Figure 8.1A Left: Convergence towards a point. Note that it is offset from the joint minimax.
Figure 8.1B Right: The familiar limit cycle. Only the short periods have been exchanged.

If the starting point is outside the realm of the limit annulus, then the trajectory swings to the edge, slides along it until it reaches the limit annulus and then enters it. This can be seen very clearly in the right side of Figure 8.1. The neatness of the trajectories in that figure is the more remarkable

when one considers that the actual Ps used in the games are $P_i' = P_i \pm \Delta P$; $\Delta P = 0.1$,so that, over one cycle, each P was varying over a fifth of its total range; refer again to Figure 7.2.

The neatness of those figures depends, of course, on the fact that they used the computed payoff, rather than the vagaries of the RNG through simulated tosses. A similar trajectory in the latter case is shown in Figure 8.2. The effect of using the RNG for simulating tosses, for short periods, means that the changes in payoff can be very large, and so also the changes in the probabilities P.

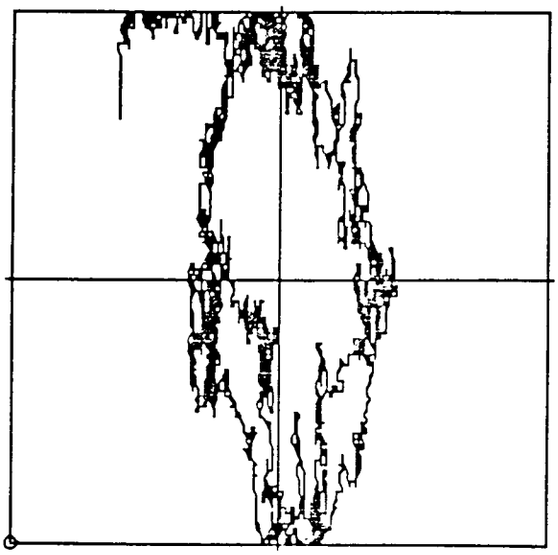
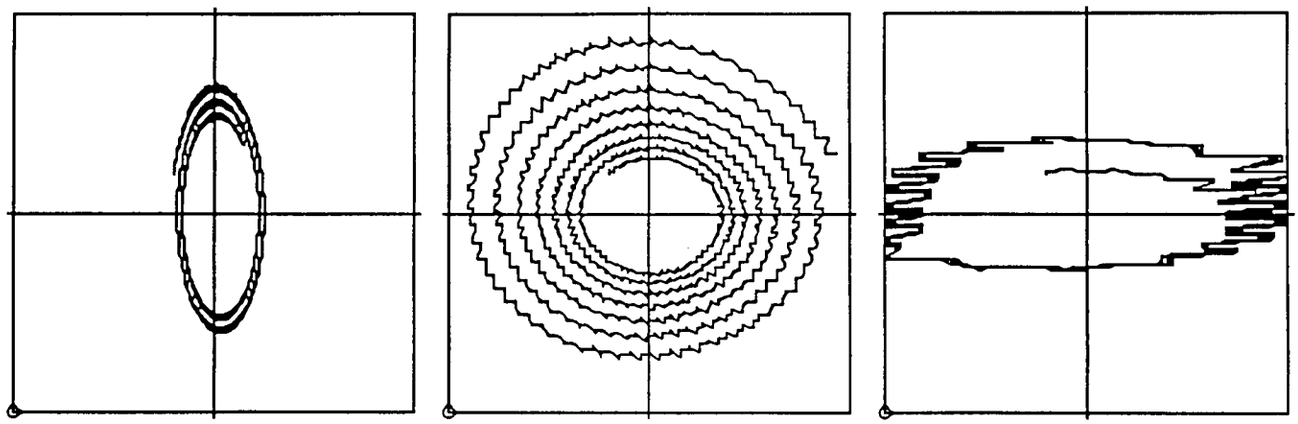


Figure 8.2 As in to Figure 8.1, but using the RNG for simulated tosses, and with smaller gains; 60,000 games

The next factor to be inspected is the gain, which changed the shape of the limit cycle in Figure 8.1 from circular to elliptical. Knowing the payoff function, and that it is continuous except at the borders, it should be noticed that it is not just the gain that matters; rather it is the product of the gain and the amplitude of the square wave; this can be confirmed by inspecting equations (3) through (6).

The sequence in Figure 8.3 shows a set of trajectories for different values of the gain G_1 of the first player.

Higher gains lead to higher gain-amplitude products, and so the corrections are correspondingly large, leading to more jagged and rough trajectories. If both gains are small, the trajectory is of course very slow and smooth: for example, Figure 8.4 shows 20,000 matches with both gains very low—0.005—and a mere three and a half cycles were made, with very slow growth. On the other hand, large gains show wild leaps and little patterning, as each correction often bumps into the stops (Figure 8.5).



$G_1 = .005$ $G_2 = .05$ $G_1 = .05$ $G_2 = .05$ $G_1 = .5$ $G_2 = .05$

Figure 8.3 Trajectories with Gains G_1 differing in order by a factor of 10; only the first one converges; in each case, 10,000 matches.

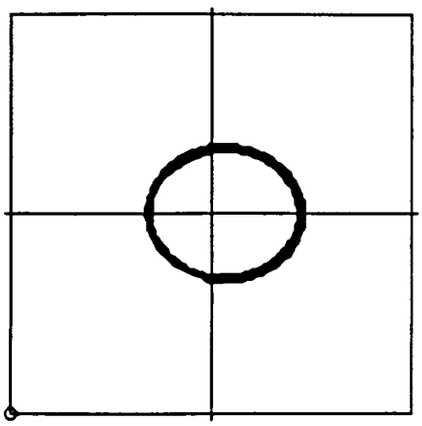


Figure 8.4 Trajectory with very small Gain

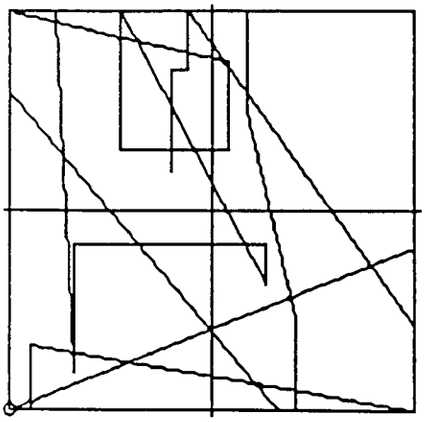


Figure 8.5 Trajectory with very large Gain

Since, as has been remarked already, it is the Gain-Amplitude product that affects the size of the corrections, and therefore changing the amplitude works very much like changing the gain. The wildness of the graph is naturally accentuated by using the RNG in simulating tosses; if we try to reproduce Figure 8.4 thereby, even with the gains set 10 times smaller, we still produce the nonconverging random walk of Figure 8.6.

The period is another matter. For longer periods, the accumulated payoff difference that is the source of the corrections will of course be much bigger, so that the trajectory will jump around very much more. Figure 8.7 shows this.

For all of those cases, the periods were approximately equal, though not exactly so; if they are not, more assymetrical trajectories can be observed, as in Figure 8.8.

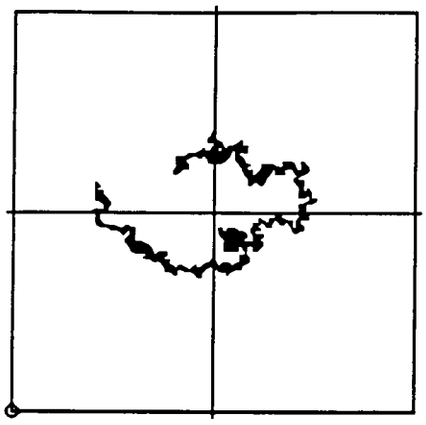
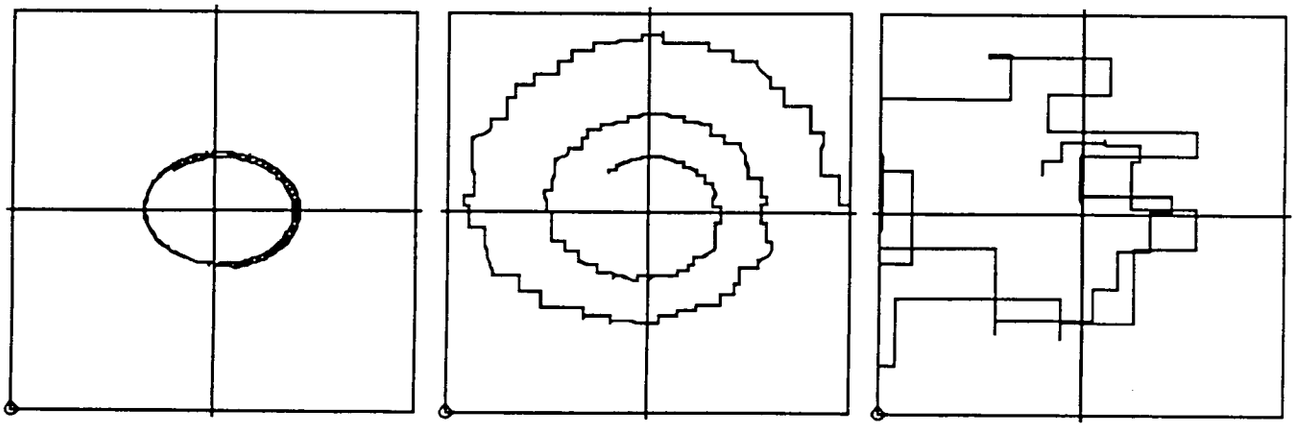


Figure 8.6 Low gains still produce random walk with simulated tosses; 80,000 matches

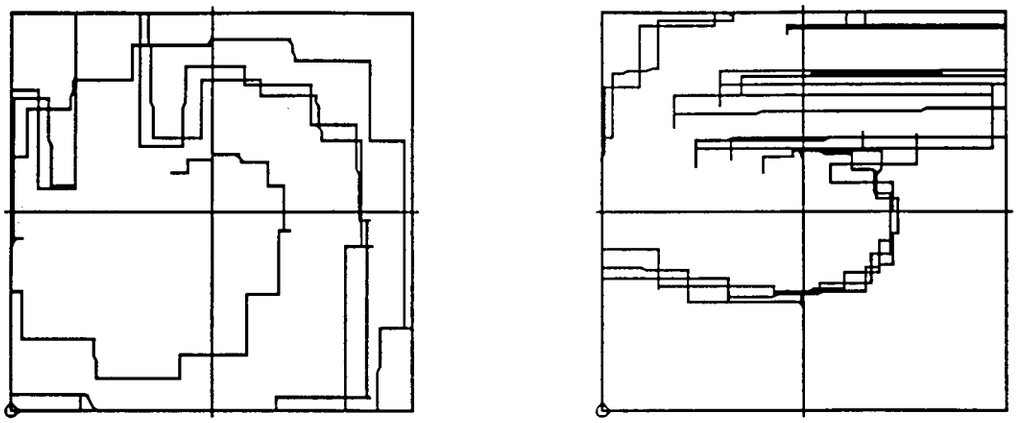


QPeriods 3,2 QPeriods 12,7 QPeriods 47,33

Figure 8.7 Effects of periods on trajectories; computed payoffs

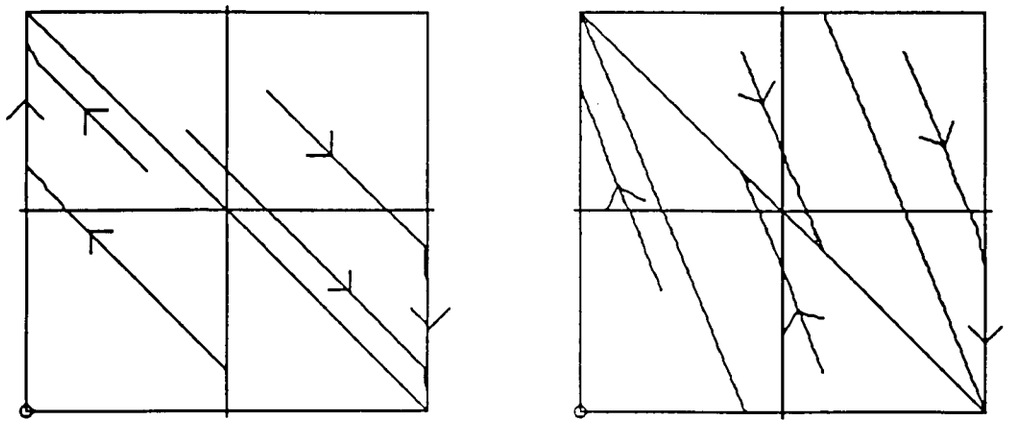
8.3 When the periods are the same

But the truly anomalous cases can be seen when both periods are the same. When all the parameters are identical, then the assymmetry of the players goals (ONE is trying to match, TWO is not) leads to a complete instability, and ONE always loses: that is, the trajectory slides to the pessimal positions for ONE, depending on the starting point. This is shown in Figure 8.9A, where the unit square is manifestly divided into two regions; they are divided by the diagonal on which there is a rolling stability; close to that diagonal, the trajectory moves very slowly.



QPeriods 3,47 QPeriods 47,3

Figure 8.8 Behavior with one short period, one long; different gains



8.9A Gains= .05, .05 8.9B Gains = .02, .05

Figure 8.9 Anomalous behavior when the periods are the same

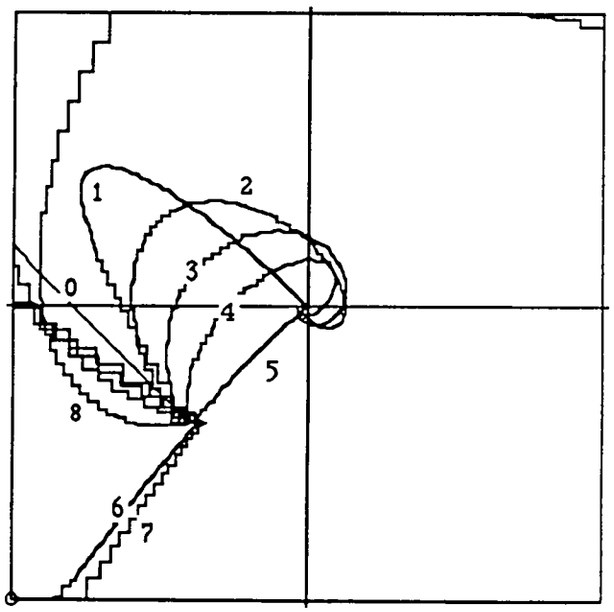
When the gains are different, but all the other parameters are the same, the figure is tipped correspondingly, as in Figure 8.9B. The dividing lines are straight at an angle whose tangent is $(G_2AMP_2)/(G_1AMP_1)$. There are two other regions, where the trajectory approaches the stable diagonal asymptotically, and stops entirely when it reaches the diagonal.

It is interesting, and obvious after reflection, that the trajectories are identical if the simulated tosses with the RNG are used. Also, the interaction between gain and amplitude are the same as before.

Now in those trials, the relative phases were the same. If there is a phase difference, the trajectories tend to one of three kinds of fixed points: solid wins for ONE, solid wins for TWO, or the joint minimax at (0.5,0.5). This is shown in Figure 8.10. The anomalies arise from the fact that, when the periods are the same, the testing changes ΔP made by one player *consistently* bias the observations made by the other.

8.4 Using the RNG when the periods are the same

There is a fair range of anomalous behaviors to be found when the periods are the same. Use of the RNG leads to the expected deviations: for example, Figure 8.11 shows a fairly exact analogy with Figure 8.9B. The long diagonal no longer serves as an asymptote, and the trajectories can leap across. Backward deviations leave the edges, always at the same angle, and the triangles at the opposite corners are inviolate.



The starting point is (0.3,0.3).
The period is 12 (QPER=3).
The fixed points for the 12 phases are:

0	(1,0)	6	(0,0)
1	(.5,.5)	7	(1,1)
2	(.5,.5)	8	(1,1)
3	(.5,.5)	9	(1,0)
4	(.5,.5)	10	(0,1)
5	(.5,.5)	11	(0,1)

Trajectories for phase differences 9, 10, and 11 overlap; for 9, it swings two corners to reach (1,0)

Figure 8.10 Anomalous trajectories; same periods, different phases

Similarly, with the RNG, the neat curves of Figure 8.10 become far untidier, even with the gains reduced by a factor of ten—see Figure 8.12.

8.5 Trajectories with numerically related periods

Even if the periods differ, there may be anomalies: perhaps one period is a multiple of the other. Figures 8.13 and 8.14 are extreme example. This arises from the biases that are caused by the uneven overlapping of the correction cycles. There seems to be little rhyme or reason in some of the shapes that can be observed. Many of the limit cycles are assymetrical or tipped, as in Figure 8.15. There are doubtless many more curiosities to be observed in this kind of phenomenon.

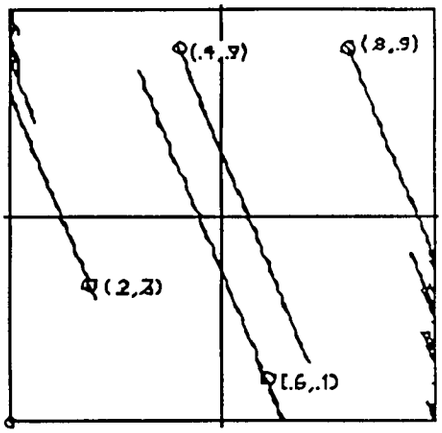


Figure 8.11 Analogy of Figure 8.9B, using simulated tosses; small circles are starting points

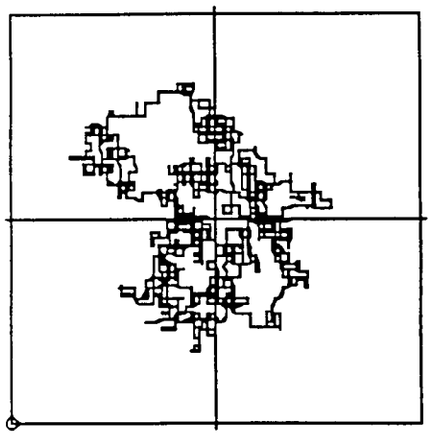


Figure 8.12 Analogy of Figure 8.10, for phase = 3, using simulated tosses

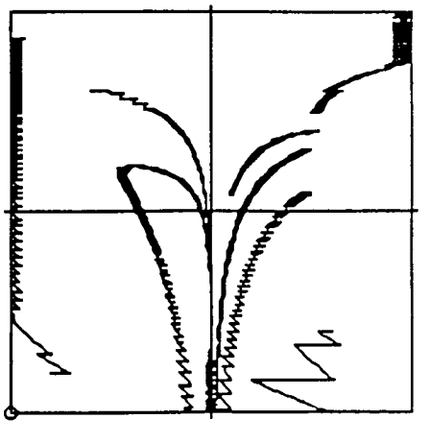


Figure 8.13 Periods 1 and 2

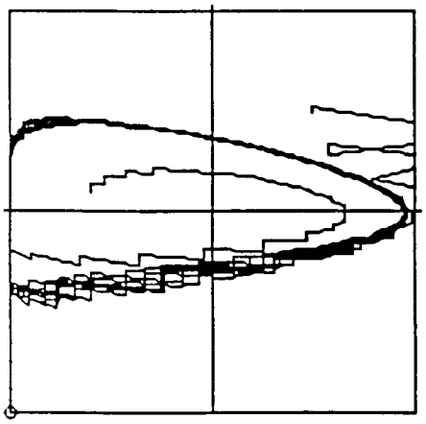


Figure 8.14 Periods 4 and 2

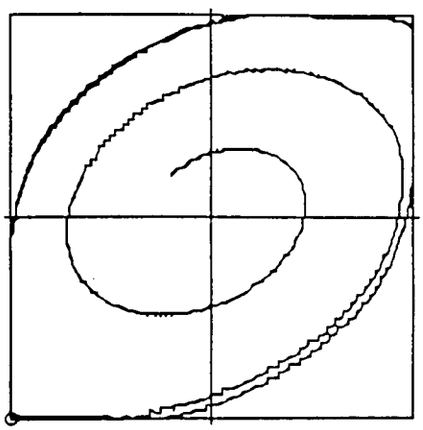


Figure 8.15 Periods 6 and 2

9.0 Adaptive *AP* Strategies or *AAP* Strategies

The enterprising game-player will notice from those examples the odd effects of the many parameters that are needed to specify precisely the *AP* strategy. He will then wonder which values of those parameters are better than other values, and in what contexts—that is, for what values of the other parameters. It will no doubt occur to him that a rational way to approach this question is to have the program adapt those values in the same way that it adapted the basic parameter, *P*.

The principles, however, are clear. It ought to be possible, the argument goes, to use identical units to control the parameters that set the underlying control mechanism. But the idea is easier to phrase than to implement. The first difficulty is that the techniques examined above control variables that take on real values; but one of the parameters of control is *PER*, which is constrained to positive integer values. This is a very important point is is considered further in Section 10.1

The second major difficulty arises from what are general problems of representation. The natural ways of representing numerical variables may be assumed—though the assumption should not be forgotten—and the question of scaling and transformations of the control parameters have to be handled. For example, should *G*, the gain, be handled directly, according to the equations, or through its logarithm? In some sense, one can argue both ways. For very small gains, the added square wave ought to be small too; which argues for a logarithmic representation; but then over some range of gains, useful convergence can very quickly drop off into instability, which suggests that the usable gain of the higher level controller has to be set very low indeed.

A third difficulty is how to express—and perhaps learn?—the useful and applicable ranges of the variables. Suppose that by some mishap of trial a probability *P* gets set to 73.5. How will the system ever recover without being able to express and also apply the knowledge that a probability lies in (0,1)?

These and other difficulties are discussed in somewhat greater detail in Section 10.

The essence of what these extended hierarchical adaptive systems can do is suggested by some of the earlier experiments. Remember that, with a single class of exceptions, there was never a trajectory where one player won more or less all the time. That exception was when both periods were either the same or closely related arithmetically. Would it be possible for continued adaptive search to seek and find that happy circumstance? The answer is yes, in some cases, but many interesting questions are raised along the path. We cannot , alas!, finish tracing that path in this paper.

10.0 Discussion and Generalizations

This section first presents a number of more or less separate topics. Many of these will stress the options available in the selection and improvement of strategies inherent in the learning atoms.

10.1 Another problem: the adapting of integer variables

In the experiments already discussed, all the variables to be controlled were continuous, even if constrained. But some of the parameters of the adaptive loop are integers; and those come in two different varieties. The first is typified by *PH*, the phase: for large values of *PER*, it can be expected that changing *PH* by 1 will not make much difference in the behavior; but for *PER*, the smallest change may make the two *PER*s equal, and that makes a drastic qualitative difference.

These two kinds of integer variables must be distinguished in any kind of machine learning. It is one of the commonest flaws in machine learning to believe that merely identifying variables as to their type is not providing the learning machine with valuable information. At some time, no doubt, the learning machine will be able to infer the type of a variable (perhaps by trial and error!); but the general problem seems to be very hard. Examples of the first kind of integer variable are:

- Any binary variable, attribute, or control;
- Example: whether a movie is in color or not; but note that a color TV program has a continuous control to change a color TV to black and white, though of course, not *vice versa*;
- Which airline to take, flying from Boston to Los Angeles;
- Whether to drink whiskey, water, or wine;
- The number of factors a natural number has.

There are examples of the other kind, that is, discrete variables *for which there is a meaningful metric*—or, for mathematicians, a relevant concept of a neighborhood:

- Amount of money, which is digital in dollars, or perhaps cents;
- The date; but note that time is a continuous variable;
- The number of light bulbs a container holds; but note that the rate of a production line making light bulbs is a continuous variable;
- The integer threshold in blackjack of whether to draw another card;
- The number of daffodil bulbs to plant for a display.

But note that all those depend on the usage to be made of the variables. In the last example, it could be reasonably supposed that the beauty of a flower bed is sort of a smooth function of the number of daffodils it contains; but if one of the requirement is that the bulbs be planted in a rectangle then the sides of the rectangle may depend on the arithmetic factors of the variable, and these are not smooth functions at all.

There is, moreover, some sort of a smooth range between discrete variables and continuous ones. Yes, it is in one sense clear that money is discrete in the number of pennies, but it is hard to image that any discussion of the national debt would turn out differently if money were in fact a

continuous value. And furthermore, there is the same kind of range between the two kinds of discrete variables.

The first class of discrete variables is represented by the periods. The interaction of the two periods is governed by their common arithmetic factors, which is a notoriously unsmooth function. In fact, another kind of adaptation has to be used, which differs in many respects from the basic adaptation technique described in section 6:

1 Run for some fixed time *T* with a given *PER*; *T* corresponds to the period for *this* adaptive loop;

2 Estimate the average or cumulative payoff, and remember it;

3 Select another value for *PER*, and run for the same *T*;

4 Estimate the average or cumulative payoff as before, and remember it;

5 Compare the two payoffs; and select as the given *PER* the one that has provided the greater payoff;

6 Return to step 3, and continue.

Note that this approach will still suffer the handicaps that have already been discussed on pp. 21-22 and shown in Figures 7.9 and 7.10.

The second class of discrete variables in this paper is exemplified by *PH*, the phase of the added square wave. As has been seen in the figures, the integer phases give some kind of a regular progression. The same adaptive procedure is used in the program, save that *AMP* is irrevocably set at 1. This represents an extra infusion of knowledge, but, as mentioned before, the difficult piece of knowledge is the *type of the variable*.

10.2 The individual controls

Problems arise with stability in many of these controls. Of course, there is nothing that says that stability is to be a goal of the strategies; which is just to do as well as possible locally. The probabilities *P* are constrained; but what are reasonable constraints for *their* controls? Perhaps *G* should be positive, but there is no inherently obvious upper limit. Indeed, if *G*—or any variable—is set at too large a value, it may enter a region of its range where change has no effect; that is exactly why *P*s are constrained to lie in (0,1), not because there is any magic knowledge.

In fact, that is a central problem of representation: If we represent a probability *P* as a real number ($-\infty < P < \infty$), so as to avoid the artificial constraints, the useful range and the useful kinds of changes remain unidentified, and it takes some special extra knowledge to put it right. In a developing and evolving system or organism, how is that knowledge to be acquired? It might strike one that that piece of knowledge is one does not fit easily into a *knowledge base*; that is another topic. In any event, in this paper we suppose that the variables have in fact been identified and typed, so that they are handled properly.

10.3 Why use a square wave?

There are many arbitrary choices in the particular strategies that form many of the details in this paper. For example, they use a square wave that is added to the control P, which tests the waters in order to estimate the gradient of the payoff function; partly for programming convenience, partly because it makes the process easily comprehensible, and partly because the same function can also be added to discrete variables.

There is one advantage in not using a square wave, but a sine wave instead—it lends an attractive susceptibility to mathematical description and analysis. One reason for this might be that the exercise of control—that is, the changing of P—might be difficult or expensive, so that the changes must be made in small inc There are several ways to set up the strategy, and one analogy to equations (6), (7), and (8) from section 6 is

$$P' := P + \Delta P = P + AMP\ sin((t + PH)/PER)$$

$$\Delta F = \int_{t-2\pi}^{t} F(P')(\Delta P/AMP)\,d\tau$$

$$P := P + G\Delta F$$

This analogy changes the time series nature of the history into a continuous one. Let us assume that the payoff function (which is a payoff *rate* in the continuous case) is locally linear, say $F = kP$. Then, with appropriate changes in variables,

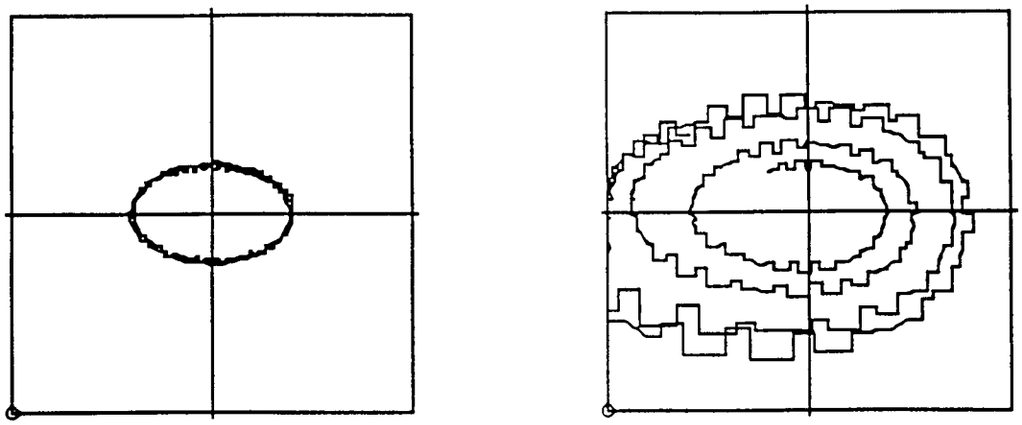
$$\begin{aligned} \Delta F &= \int_0^{2\pi} kP'\ sin\ t\ dt \\ &= \int_0^{2\pi} k(P + sin\ t)\ sin\ t\ dt \\ &= \int_0^{2\pi} k\ sin^2\ t\ dt = \pi k \end{aligned}$$

and P is changed in the correct way, so as to drive it towards the value that maximizes F. Of course, the arguments from section 7.3 will mean that the integration probably should be taken over the interval $(\pi/2, 5\pi/2)$. Figure 10.1 shows a comparison of trajectories from square waves and sine waves.

10.4 Local reasonableness and applications

We can refer to these strategies as *locally reasonable*, in the sense that they attempt to climb the observed gradient:

- The adaptive strategy makes the kind of change that is responsive to a locally measured difference or gradient in the payoff function.



Added sine wave Added square wave

Figure 10.1 Trajectories with different added testing functions. The square wave provides much bigger average differences

- The observed gradient in the examples here is merely a two-way comparison, *made sequentially*;
- The observed gradient is measured with respect to an environment that is often changing; it has already been shown that a direct arithmetic difference should not be used straightforwardly—this was discussed also in section 7.1.
- This phenomenon becomes worse in environments of higher dimensionality.

That argument can be restated in the context of a particular business application:

- A corporation XYZ sells a product at a price Q. The marketing Department sets Q by comparing revenues (or perhaps profits from revenues) during recent periods in which Q was set at different values.
- Supposing that the market is rising; that is, that customers are increasingly willing to pay for the product, and even pay a higher price. Then XYZ's policy will tend to favor *any recent change* it has tried. And *vice versa*; if the market is falling; its policy will tend to want to reverse any recent change.
- There is a competitor YLE, who, by clever analysis or possibly industrial espionage, has ascertained the market testing and Q-setting policy of XYZ. YLE follows an *AAP* strategy, and makes sure that every time XYZ raises prices, it makes *more* money; every time it lowers them, *less*. YLE can then drive XYZ into bankruptcy (Figure 10.2) and corner the market, unless XYZ changes its policy; the point is that XYZ can reasonably believe that it is undertaking the right changes. For, by our terminology, XYZ and its price-setting office is certainly following a *locally reasonable* policy. The effect from the point of view of company XYZ is shown in Figure 10.2.

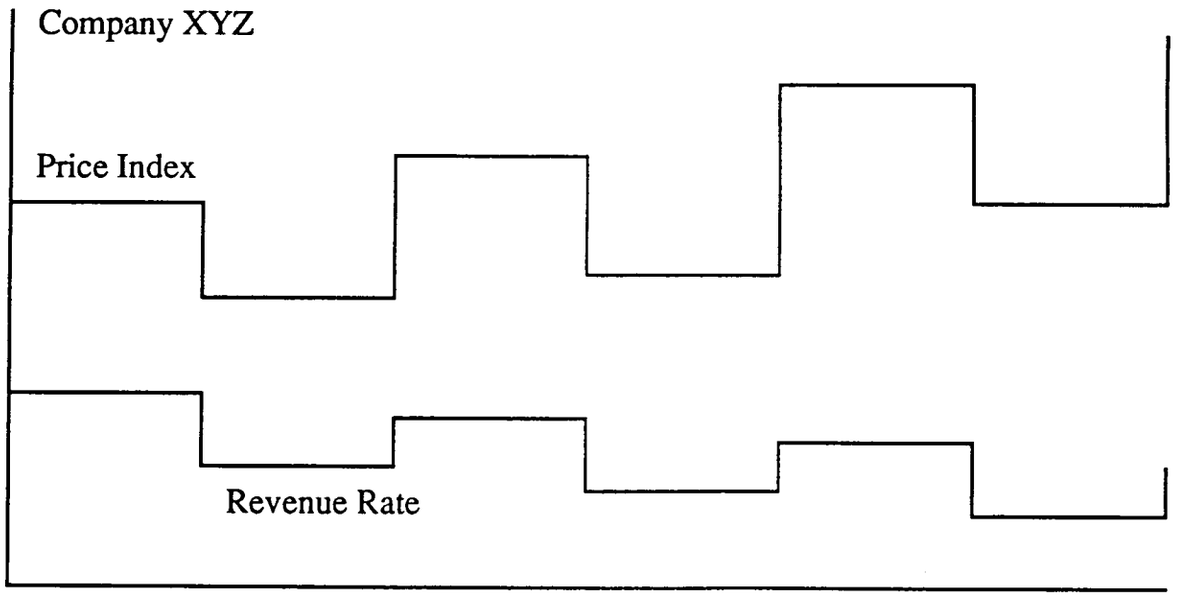


Figure 10.2. Every time Company XYZ lowers prices, revenues fall. Every time it raises prices, revenues rise. What would you do?

Notice that this discussion refers to a competition that is not zero-sum; not only that, but it is not even two-person—for the two companies also have a third player, the customers.

10.5 Continual Feedback

In all the previous examples, the control—that is, the direct changes in P—were made once every full cycle. It is clear that that is not required by the concept, and that, especially if the changes are small, they can be made at every step. Examples are shown in Figure 10.3 These figures are a little busier than the previous ones, as a result of the plotting: a point (P_1,P_2) is plotted only when one of the two Ps changes; and in these figures, they change at each move. But the general clockwise rotation of the trajectory is the same.

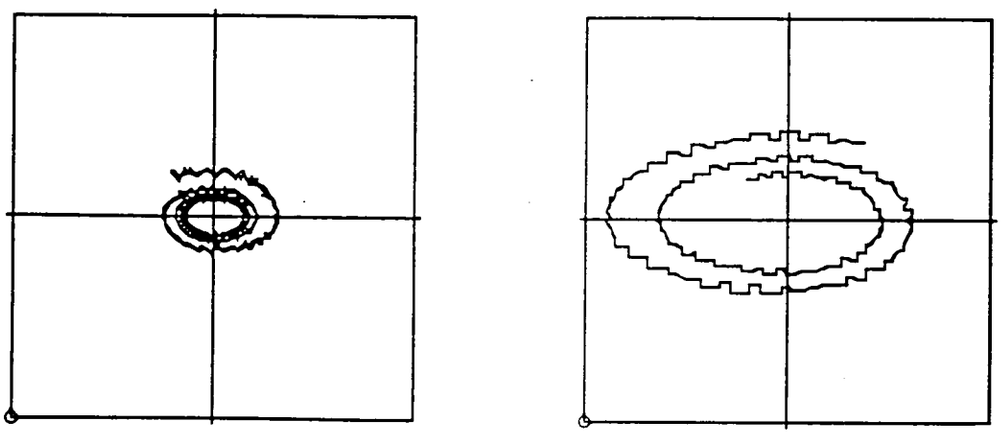


Figure 10.3 Continual correction (left) vs. periodic correction (right). Note that the continual correction causes convergence

10.6 Linearity

The actual changes made to the probabilities P are linear functions of the cumulative payoffs—or the immediate ones, as in the preceding subsection—as in equation (6): $P' := P + G\,\Delta F$. That is of course an arbitrary choice. Different functions do not change the overall picture much, save that they should be monotonic in the appropriate direction. Figure 10.4 shows one example using

$$P' := P + (\Delta F)^3$$

where the other parameters of the program are the same as in Figure 10.3.

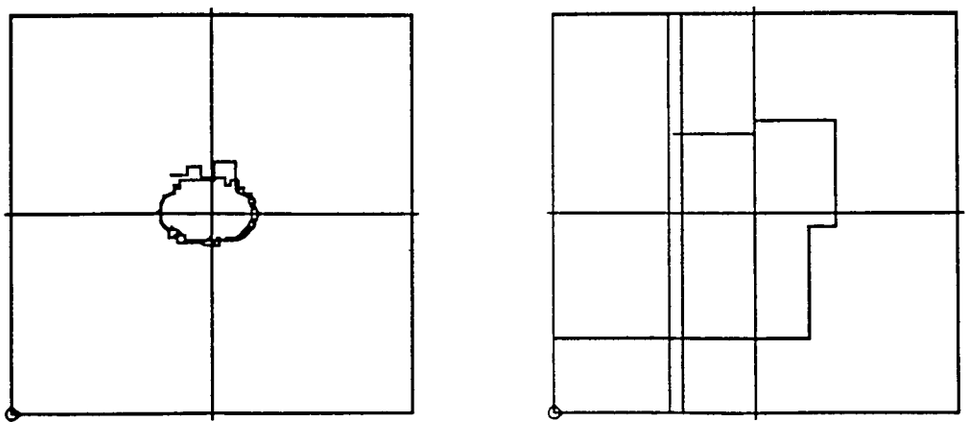


Figure 10.4 Correction proportional to the cube of the difference; notice the difference made by the starting point; on the left it is (0.4,0.6), and the right, (0.3,0.7)

This makes the trajectory very sensitive to the closeness to the minimax, since differences in payoffs encourage stability when they are small, but diverge quickly when they are large. In the figure, the left trajectory started at (0.4,0.6), and then slowly converges; but the right trajectory, starting only twice as far away from (0.5,0.5), diverges instantly and never recovers.

Moreover, it is not an essential part of the adaptation that the square wave be *added*; any kind of perturbation will suffice, providing only that the relative payoffs are appropriately assigned. Figure 10.5 shows what happens when the gradient is detected by multiplying the Ps by 1.1, say, and then dividing them.

10.7 Limitations

There are large limitations indeed in what such strategies can accomplish:

- Their basic vocabulary of actions and senses is extraordinarily limited. Suppose one player is persistently alternating his moves by, say, choosing heads with alternate probabilities 0.4 and 0.6; the other player has no way to detect that. For that matter, how sure are we that we could detect it?

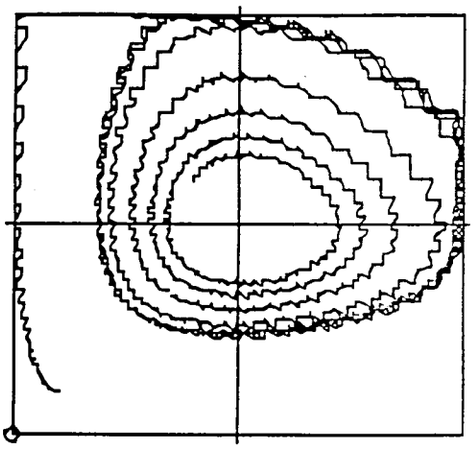


Figure 10.5 Using multiplication instead of addition for determining the gradient; notice the assymmetry of the limit cycle; starting points are (0.4,0.6) and (0.1,0.1)

- The evaluation function is here very simple, with one explicit aspect, the payoff function, and an implicit one, the period. The latter sets up a kind of *time constant*, so that the strategy's memory of its experiences is very limited; it's rather like a Markov process. In real systems, not only would the evaluation function be more complex, but it would also include many other elements; it might cost something to change P; or cost more to make a larger change than a smaller one, for example.
- Now the natural (might one say *human*?) ways to approach the good strategy are very different from those presented above; which are essentially 'brute force' or 'exhaustion' adaptation. One might analyze the opponent's strategy directly; for example, a simple statistical test would determine both the period and the phase of the added square wave, and the correct values for the appropriate strategy would follow immediately.
- There is no easy way at this level to take into account other forms of knowledge; especially those that in AI are most easily represented *symbolically*; this of course is a large and central problem in machine learning.

11.0 Conclusions

There are some general conclusions from checking these results and many others like it:

- *Convergence to stability with a random element is far noisier than with the computed payoffs, and, if it does occur, it is far more likely to form a "blob of accumulation" than a "point of accumulation."*
- *Small changes in parameters can make big differences in the competitive outcome.*
- *Two units that are competing for some resource do not together reliably exhibit stability in their joint behavior.*

Units, systems, or programs may exhibit admirably robust behaviors in searching for maxima of functions, tracking moving variables, and so on. Individually, the very general gradient detection mechanisms described here do so. But the interactions between them when they are in effect

competing for some resource vitiates that assurance of stability. This observation is applicable, for example, to many current connectionist systems; where the so-called "neuron" units may be competing, perhaps through some back-propagation procedure, for additions to their weights.

Furthermore, even if some configuration is stable and convergent, some slight changes in its parameters, or in the loads or tasks assigned, may destroy that stability.

The suggestion or implication is that additional structure is needed. There is nothing that can be drawn from this work that states what ought to be the nature of that structuring or even where it should be placed. An obvious extension is to consider some straightforward form of hierarchical control. Such studies are handicapped by the fact that almost no work has been done on any kind of general hierarchical control; but in principle the programs to extend this work to that problem are not difficult, and they would provide a testbed for the problems and difficulties discussed in Section 10.

This lack is aggravated by the extreme simplicity of the model, and the assumption that a single evaluation function or "purpose" is driving each unit. In effect, when one of the players in *PM* changes his period, he is changing his evaluation function—altering the time constant of what is to be considered reward. But that does not extrapolate easily to other forms of alteration of purpose; let alone to multiple purposes.

BIBLIOGRAPHY

[BAR84] Barnett, J.A., "How Much is Control Knowledge Worth? A Primitive Example," *Artificial Intelligence*, **22** (1984), 77-89.

[HAG56] Hagelbarger, David, "SEER, a SEquential Extrapolation Robot," *IEEE Transactions on Electronic Computers, EC 5 #1*, March 1956.

[OGS89A] Selfridge, O. G., "Adaptive Strategies of Learning: A Study of Two-Person Zero-Sum Competition," *Proc. 6th Int'l Workshop on M.L.*, Cornell, Morgan Kaufmann, 1989, pp. 412-416.

[OGS89B] Selfridge, O.G., "Atoms for Learning," in preparation.

[VonN48] Von Neumann, J., and Morgenstern, O., *Theory of Games and Economic Behavior*, John Wiley and Sons, New York, 1946.

[WIN88a] Windecker, R.C., "Learning in Networks of Nondeterministic Adaptive Logic Elements," in *Neural Information Processing Systems*, Ed. Anderson, D.Z., American Inst. of Physics, New York 1988, 840-849.

[WIN88b] Windecker, R.C., "A Class of Nondeterministic Adaptive Automata that Play Matching Pennies," AAAI Spring Symposium Series, March 1988.

APPENDIX: The program and the parameters for the Figures

The programs that produced the figures and the data were written in Microsoft[tm] QuickBasic[tm] for Apple[tm] Macintosh[tm] Systems, and run on a Mac II.

For each of the computer-generated figures, the parameters are provided in order: initial values of the probabilities, the starting point (SP); gains (G); amplitudes (AMP); quarter periods (QP); phases (PH); whether the program used computed payoff function (CP) or simulated tosses (RNG); and the kind of display, history plot (HP) or trajectories (TR). PH data are often omitted, especially when the QPs differ.

Figure	SP	G	AMP	QP	PH	Type	Display
7.1	.4,various	.0001,-	.05,-	30,-	-	RNG	HP
7.2	.4,.75	.01,-	.1,-	43,-	-	RNG	HP
7.3	.4,.75	.001,.001	.1,.1	30,43	-	RNG	TR
7.4	.4,.75	.001,.001	.1,.1	30,43	-	RNG	HP
7.5	.4,.75	.001,.001	.1,.1	30,43	-	CP	TR
7.6	.4,.75	.001,.001	.1,.1	30,43	-	CP	HP
7.7	.4,.6/.1,.9	.03,.012	.05,.05	5,7	-	CP	TR
8.1A	.2,.8	.01,.05	.1,.1	2,3	-	CP	TR
8.1B	.2,.8	.01,.05	.1,.1	3,2	-	CP	TR
8.2	.2,.8	.001,.005	.1,.1	2,3	-	RNG	TR
8.3	.4,.6	several, .05	.1,.1	3,2	-	CP	TR
8.4	.4,.6	.005,.005	.1,.1	3,2	-	CP	TR
8.5	.4,.6	.5,.8.	.1,.1	3,2	-	CP	TR
8.6	.4,.6	.0005,.0005	.05,.05	3,2	-	RNG	TR
8.7	.4,.6	.02,.015	.05,.05	various	-	CP	TR
8.8	.4,.6	.02,.015	.05,.05	3,47/47,3	-	CP	TR
8.9A	various	.05,.05	.05,.05	5,5	0,0	CP	TR
8.9B	various	.02,.05	.05,.05	5,5	0,0	CP	TR
8.10	.3,.3	.05,.05	.05,.05	3,3	various	CP	TR
8.11	various	.02,.05	.05,.05	5,5	0,0	RNG	TR
8.12	.3,.3	.05,.05	.05,.05	3,3	3	RNG	TR
8.13	various	.5,.15	.05,.015	1,2	0,0	CP	TR
8.14	various	.5,.15	.05,.015	4,2	0,0	CP	TR
8.15	various	.01,.015	.08,.06	6,2	0,0	CP	TR
10.1	.4,.6	.02,.024	.05,.015	9,13	-	CP	TR
10.3	.4,.6	.02,.024	.05,.015	9,13	-	CP	TR
10.4	.4,.6/.3,.7	02,.024	.05,.015	9,13	-	CP	TR
10.5	.4,.6	.05,.05	special	3,2	-	CP	TR