ELSEVIER

Journal of Economic Dynamics and Control
20 (1996) 173–207

JOURNAL OF
Economic
Dynamics
& Control

Finite automata play repeated prisoner's dilemma with information processing costs

Teck-Hua Ho

Anderson Graduate School of Management, University of California at Los Angeles, Los Angeles, CA 90024-1481, USA

(Received May 1992; final version received December 1994)

Abstract

We study the finitely repeated Prisoner's Dilemma game. Our players are modeled as finite automata. A population of boundedly rational players compete in a 'survival of the fittest' evolution contest simulated using Holland's genetic algorithm. Starting from a hostile population which plays defection frequently, our simulation results show that players converge to play cooperatively. This emergence of cooperative behavior breaks down when we penalize a complex strategy based on the size of the machine. On the other hand, a penalty cost that increases with the frequency an automaton switches states will not hurt the development of cooperative behavior.

Key words: Prisoner's dilemma; Genetic algorithm; Learning; Computational complexity
JEL classification: C73

1. Introduction

1.1. Bounded rationality in game theory

The concept of rational behavior is frequently used in game theory. A player is *rational* if she makes decisions consistently in pursuit of her own objectives. In

I am very grateful to John Miller for his suggestions in the design of the genetic algorithm software and his constant encouragement during the period this research was undertaken. John also provided a software routine for computing the parameters of Moore machines. Comments from Colin Camerer, Eric Johnson, Paul Kleindorfer, Mark Knez, James Laing, and participants at the international conference of game theory at Florence, Italy were helpful. I thank Taizan Chan for his able research assistance.

SSDI 0165188994000848 C

game theory, each player's objective is to maximize her expected utility. This rationality assumption is an idealization about players and is often needed for computing equilibria. The term 'bounded rationality' is used to describe human choice that takes into account the cognitive limitations of the decision-maker (Simon, 1982). Since the beginning of 1980s, the 'bounded rationality' problem became an important research area in game theory. Now there is a consensus that future development of the field relies critically on finding a satisfactory way of modeling bounded rationality[1] (Binmore, 1987, 1988; Aumann, 1989; Kreps, 1990; Selten, 1991).

There are three approaches to modeling bounded rationality in game theory. The first approach develops notions of complexity of a strategy by representing players' strategies as finite automata or Turing machines. This approach essentially turns the traditional unconstrained optimization problem into a constrained one, confining players' choices of strategies to those that are implementable by machines. The approach allows us to attach cognitive costs to strategies similar to the way we attach costs to physical factors of production in the theory of firms (see Neyman, 1985; Radner, 1986; Rubinstein, 1986; Abreu and Rubinstein, 1988; Kalai and Standford, 1988; Zemel, 1988; Gilboa, 1988; Banks and Sundaram, 1990).

The second approach stresses Maynard Smith's (1982) notion of evolutionary stable strategies (ESS).[2] Under this notion, players are not required to find maxima; they are picked by the evolution process. The advantages of this approach are that it rests on the principles of natural selection and it can produce results that have strong implications (see Selten, 1983; Binmore, 1987, 1988; Aumann, 1989; Fudenberg and Maskin, 1990; Binmore and Samuelson, 1990; Friedman, 1991).

The third approach places each individual player in a dynamic context and assumes that she is a myopic and adaptive learning agent. At any point in time, she is assumed to possess a partial/simplified model of her environment. She then chooses an optimal action within the framework of the model. As she gathers more information about the environment, she adjusts her model accordingly so that it becomes closer to the actual environment. Usually, the adjustment procedure is some reasonable heuristic procedure (see Fudenburg and Kreps, 1990; Milgrom and Roberts, 1991). The classical Cournot dynamics and Brown's (1951) fictitious play are two early examples.

Like the above three approaches, we do not assume the rational hypothesis in this paper. Our players are not unconstrained utility maximizers who possess

---

[1] Some believe that the multiplicity of equilibria problem can only be solved by modeling bounded rationality (Kreps, 1990).

[2] An ESS is a strategy such that, if all of the members of a population adopt it, then mutant strategy could not invade the population and thrive under natural selection.

infinite computational resources; they are limited in cognitive ability. Like the first approach, we model bounded rationality using finite automata. However, our players are myopic; they cannot contemplate all future plays from the beginning of play. Our game outcomes are results of iterated play by myopic players. In this sense, our approach is similar to the third approach. Specifically, it is assumed that players learn according to Holland's genetic algorithm.

1.2. Related research and contributions

This research is closely related to Axelrod (1984, 1987) and Miller (1989). Axelrod (1984) conducted two computer tournaments for a finitely Repeated Prisoner's Dilemma (RPD). In the first tournament, 14 game theorists each contributed a computer program to play the RPD. The 14 programs and a totally random strategy were paired together in a round robin tournament and 200-round RPD games were played. The programs were ranked according to the total payoff accumulated. The winning program was also the simplest. It was Tit-For-Tat. The results of the first tournament were circulated and entries were solicited for a second tournament. Sixty-two contestants from six countries submitted their programs. This time the games played were not of exactly 200 rounds, but were of random length with median 200. Again, Tit-For-Tat was the winner.

Axelrod (1987) later applied Holland's genetic algorithm to evolve RPD strategies against a fixed environment which consisted of eight representative programs from the two computer tournaments. From a randomly selected population, the algorithm produced a strategy that was as successful as Tit-For-Tat, the winner of the two computer tournaments. In addition, the algorithm selected a strategy that performed substantially better than Tit-For-Tat if it would start from a population whose rules were similar to Tit-For-Tat.

Miller (1989) integrated the notion of finite automata with Holland's genetic algorithm, and investigated the development of cooperation in a self-evolving environment. A population of individuals plays the RPD against each other in a round robin basis, and the population evolves continuously. Hence, the environment used is a variable one. He found that players exhibited cooperative behavior and attained payoffs which are close to the pareto optimal payoff. He also studied the effect of imperfect information on the development of cooperation, and found that less accurate information could easily lead to a lower level of cooperation. This is because a cooperative play that is misrepresented because of imperfect reporting may start a series of retaliation and counter-retaliation between the players.

Both Axelrod and Miller study an environment that is relatively conducive to the development of cooperative behavior. Axelrod uses an environment consisting

of eight representative programs from the computer tournaments; many of these programs were variants of Tit-For-Tat. Thus, the environment mainly consists of cooperative individuals. Miller uses an initial environment that cooperates and defects randomly. Each individual adopts a 50–50 mixed strategy. We extend this stream of research by investigating whether cooperation can emerge in a 'tough' and evolving environment where players are hostile and defect frequently.

In addition, we study complexity of strategy in a dynamic context. We use the size of an automaton and the frequency with which it switches states during the play of a game to measure complexity of strategy. The size of an automaton is the minimum number of accessible states of the machine. Rubinstein (1986), Abreu and Rubinstein (1988), and Kalai and Stanford (1988) also use the size of an automaton to measure complexity. While the size of a machine is an intrinsic parameter, the number of times a machine switches states depends on the nature of opponents the machine faces. Banks and Sundaram (1990) use the frequency of switching to study complexity. Linster (1992) uses both to measure complexity.

The complexity measures have some simple cognitive interpretations. The size of machine can be interpreted as long-term 'storage' cost. It is the cost of remembering the states that make up a strategy. A more complex strategy takes up more storage space. Note that the size of machine is a good measure for the cost of monitoring. The frequency of switching can be interpreted as the frequency of retrieving information from long-term to short-term memory. Since each retrieval of information is costly, strategies that switch more frequently are more complex. Under this framework, we can control the degree of bounded rationality by varying the amount of fee charged for a complex strategy. In so doing, we contribute to the current understanding of bounded rationality by providing insights into the role of complexity of strategy in evolution of cooperative behavior in RPD. In particular, we show that cooperative behavior may break down if players are averse to complexity.

1.3. Finite automata and Holland's genetic algorithm

In the next few paragraphs, we describe what finite automata and Holland's genetic algorithm are, and discuss their promises as tools for modeling bounded rationality in repeated games.

A finite automaton is a kind of dynamic system that changes its behavior only at the discrete moments of time under consideration (see Minsky, 1967, and Hopcroft and Ullman, 1979, for an introduction). The system consists of a finite set of internal states (one of which is the initial state), an output function, and a transition function. The output function determines the output of the system as a function of the state. The transition function determines the next state of the system as a function of the input and the current state of the system. Thus, the

system begins in its initial state and gives the output specified in that state. Depending on the input, the system moves to a new internal state. This process can continue forever or for a finite number of steps.

The use of finite automata to model bounded rationality in the context of repeated games was first suggested by Aumann (1981). Since then, several researchers have employed finite automata to model bounded rationality and have produced interesting results. For example, Neyman (1985) and Zemel (1988) show that mutual cooperation in every round in a finitely RPD can be a Nash equilibrium if players' strategies are modeled as finite automata. Rubinstein (1986) and Abreu and Rubinstein (1988) show that the set of equilibrium payoffs in an infinitely RPD[3] is more restrictive if players seek to minimize the complexity of their strategies represented as finite automata.

We simulate players' myopic learning behavior by Holland's genetic algorithm. 'Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search' (Goldberg, 1989). Using a genetic algorithm, one represents strategies (finite automata) as string structures. Each string structure serves a dual purpose: it provides a representation of what the strategy will become, and it also provides parts that can be transformed to yield new strategies for the next generation.

The genetic algorithm is adopted in this research for several reasons. First, it has been used successfully to model learning economic agents in complex economic settings (Marimon, McGrattan, and Sargent, 1990; Marimon and Miller, 1989; Holland and Miller, 1991) and to evolve strategies in games (Axelrod, 1987; Miller, 1989). Second, the algorithm is a powerful search algorithm and has been shown to work in difficult domains. Holland suggests that the power of the algorithm comes from the 'implicit parallelism' of search (Holland, 1975). Third, the algorithm is central to the integrative framework of Holland, Holyoak, Nisbelt, and Thagard (1986) which has been used successfully for explaining human adaptive behavior. The framework has been shown to be able to treat a variety of empirical evidence from conditioning, concept formation, and problem solving to scientific discovery. Fourth, it has a direct mapping to evolutionary dynamics which makes the behavior of the algorithm easy to interpret.

---

[3] In an infinitely RPD or a PD that is repeated with a finite but unknown number of times, many outcomes can theoretically emerge as equilibrium outcomes. In fact, the Folk Theorem states that any indvidually rational payoff vector can be the outcome of a perfect equilibrium if payoffs are calculated as 'the limit of the mean' and players do not discount the future too much (Fudenberg and Maskin, 1986).

2. The repeated Prisoner's Dilemma, finite automata, and genetic algorithm

2.1. The repeated Prisoner's Dilemma

The Prisoner's Dilemma is a well-known 2×2 game created by A.W. Tucker. The game has been used to model important economic and political problems such as oligopolistic collusion, international trade, arms race, and public goods provision. The payoff matrix below (Fig. 1) is a typical Prisoner's Dilemma game. If Row and Column cooperate, they each get R; if both defect, they each get P. However, if one cooperates but not the other, the cooperator gets S, while the defector gets T.

Three arguments support playing defection in this game:

(i) It is the only Nash equilibrium: For any other pair of pure or mixed strategies at least one player has an incentive to change his strategy, given that the other's strategy is fixed.
(ii) It is the maximin strategy: Defection guarantees at least P, cooperation might give S ($S < P$).
(iii) Defection strictly dominates cooperation: Regardless of Row's (Column's) strategy, Column (Row) is always better off if he or she chooses defection because ($T > R$ and $P > S$).

Note that argument (iii) implies (i) and (ii) weakly. Argument (iii) is very compelling because it means that each player's best response is independent of what the other player does (the so-called Sure-thing Principle). Similar results hold if the game is iterated a known fixed number of times in a supergame. The supergame has defection at every stage as the only perfect equilibrium outcome.

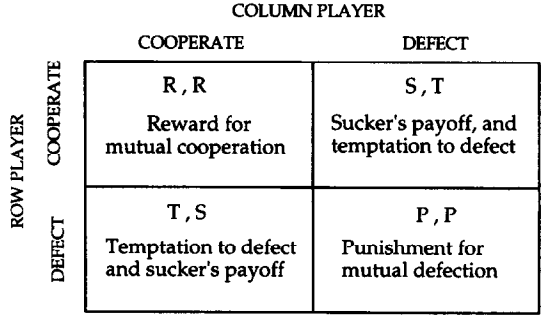


Fig. 1. Prisoner's Dilemma.

An iterated dominance or a backward induction argument also produces mutual defection in every round as the only equilibrium outcome.

The mutual defection solution is unsatisfactory for three reasons. First, there are a lot of empirical data that contradict it (e.g., Selten and Stoecker, 1986; Rapoport and Chammah, 1965). Second, the solution is Pareto inferior; (R, R) strictly payoff dominates (P, P). Third, the solution is very sensitive to any slight change in the game form. For example, Kreps et al. (1982) show that if players have the slightest doubt in their opponents' rationality, cooperative behavior can emerge.

As indicated above, cooperative behavior can emerge in the RPD if 'bounded rationality' is incorporated (Neyman, 1985; Axelrod, 1987; Radner, 1986; Miller, 1989). We test the robustness of this result with respect to the toughness of the initial environment and the manner in which 'bounded rationality' is captured. 'Bounded rationality' is captured here by charging a higher fee for using a more complex strategy. Thus our paper provides some 'stress tests' of the maintained hypothesis that cooperative behavior can emerge in RPD if players are boundedly rational.

2.2 Finite automata

Finite automata are finite-state machines that do not have external memory. Finite-state machines that have an external memory are called Turing machines. Turing machines are very powerful and versatile. In fact, the Church–Turing thesis asserts that any formal calculation possible for a human mathematician can be mimicked by a Turing machine.[4] Some economists use Turing machines instead of finite automata to model bounded rationality (Binmore, 1987, 1988; Anderlini, 1989). Fig. 2 is a schematic of different levels of players' complexities that have been assumed by various economists.

A finite automaton whose output depends on only the internal state is called a Moore machine. Formally, a Moore machine is a quadruple $\langle Q, q_0, \lambda, \delta \rangle$, where Q is a finite set of internal states, $q_0 \in Q$ is the initial state, λ: $Q \to S_i$ is the output function, $S_i \in \{C, D\}$ is player i's move in the current period, and δ is the transition function which maps the current internal state of the machine and the reported move of the opponent into the next internal state, i.e., δ: $Q \times S_{-i} \to Q$ ($S_{-i} \in \{C, D\}$ is the opponent's move in the current period.) Thus, a Moore

[4] It is called a thesis because there is no mathematical proof for it, but nobody has been able to find a counter-example to the thesis. The traditional approach to game theory implicitly assumes that each player is better than a Turing machine since a Turing machine always implements a function that is effectively computable, but a player in traditional game theory is assumed to be able to compute any function. Roughly speaking, an effectively computable function is one which could possibly be computed by any imaginable finite device.

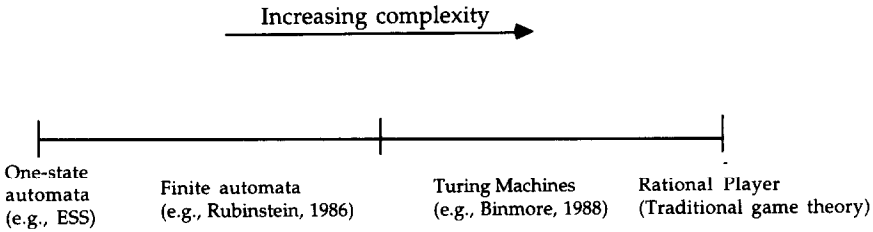


Fig. 2. A schematic of different levels of complexity.

machine begins in its initial state and performs the action specified in that state. Based on the reported move of the opponent, the machine moves to a new internal state (including the current state). This process continues until the game ends.

The concepts of Moore machines and how they represent strategies are best illustrated by their transition diagrams. Fig. 3 shows some typical RPD strategies and how they are represented by the Moore machines. The nodes of the transition diagrams represent the internal states, and the letter in parentheses indicates the move of the machine when it enters that state. The node q_0 is the starting state and has an arc labeled S pointing to it. The output function is captured in the correspondences between the internal states and the associated moves. The transition function is represented by arcs between the nodes. The letter labeling the arc is the reported move of the opponent, i.e., the transition is dependent on the opponent's move.

In Fig. 3, the first machine cooperates constantly. The second machine defects constantly, i.e., a Nash equilibrium strategy. The third machine is the Trigger strategy: it starts with cooperation, and switches to and continues with defection once the opponent is reported to defect. The fourth machine is the Tit-For-Tat strategy. The strategy starts by cooperating in the first period and thereafter follows whatever the opponent does. The fifth machine is the Punish-Twice strategy. It punishes every defection by the opponent with two consecutive defections in a row. The last machine is the Forgive-Once, Punish-Once strategy. It punishes the opponent once when it defects two rounds in a row. These machines can be characterized by the following intrinsic parameters (Miller, 1989):

n = size of a minimal machine,[5]
cr = cooperation reciprocity,

[5] This is to be distinguished from the size of a nonminimized Moore machine and the cardinality of Q (Harrison, 1965). The size of a minimal machine is equivalent to the cardinality of the induced strategy set (Kalai and Stanford, 1988), i.e., the numbr of distinct strategies induced by the original strategy in all subgames. Kalai and Stanford define the complexity of a strategy in a given repeated game to be the cardinality of the induced strategy set. For example, in Fig. 3, trigger strategy induces itself if the opponent cooperates and induces an always defect strategy if the opponent defects. Hence it has a complexity of 2, which is the same as the size of the minimal Moore machine.

dr = defection reciprocity,
ts = proportion of terminal states.

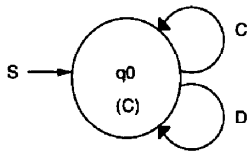
In addition, we define the following two variables which depend on the opponent the machine is playing against during the play:

fs = frequency of switching during the play of the game,
fc = frequency of playing cooperation during the play of the game.

In Fig. 3, n is computed by counting the number of accessible states (since they are all minimal machines); cr is the proportion of cooperations in response to cooperations by the opponent in the preceding rounds; dr is the proportion of defections in response to defections by the opponent in the preceding rounds; ts is the ratio of the number of terminal states to the size of machine where terminal states are absorbing states that do not switch to other states regardless of the opponent's move. The values of cr and dr are calculated based on the assumption that all states are equally likely to occur. The values of fs and fc shown are computed based on a ten-round repeated game and each of the machines is playing against an opponent whose strategy is [C, C, C, C, C, D, D, D, D, D] i.e., who cooperates in the first five rounds, but defects in the last five rounds. For example, consider the Punish-Twice strategy. As represented, the machine is a minimal machine and hence n is 3. cr is 0.66 because the machine reciprocates two of three Cs (the three arcs that are labeled C) played by the opponent. Similarly, dr is 0.66 because the machine reciprocates two of the three Ds played by the opponent. There is no terminal state, and therefore ts is $0/3 = 0$. In response to a strategy which cooperates in the first five rounds and defects in the last five rounds in a ten-round repeated game, the machine makes the following transitions [q_0, q_0, q_0, q_0, q_0, q_0, q_1, q_2, q_0, q_1, q_2] and the following moves [C, C, C, C, C, C, D, D, D, D]. Hence fs is $5/10 = 50\%$ and fc is $6/10 = 60\%$.

The above discussion suggests that Moore machines provide a parsimonious representation of a wide range of strategies. However, there is a number of strategies that Moore machines do not capture depending on their size. For example, consider a sixteen-state Moore machine. All the strategies discussed above can be individually represented by the machine. The sixteen-state machine however cannot represent a strategy that counts the number of cooperations during the course of the game if the number of rounds of the repeated game is more than sixteen rounds. Such a strategy will outperform either Trigger or Tit-For-Tat strategy because the strategy can count the number of cooperations up to the round before the last round, and defect in the last round to obtain a higher payoff. It has been shown that if the size of machine is exogenously fixed, then Tit-For-Tat can arise as optimal play for a finitely RPD as long as the number of rounds of the RPD is larger than the size of machine used to play the game (Neyman, 1985).

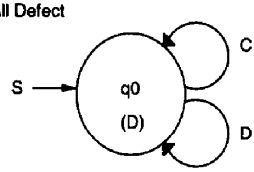
All C



Parameters and variables

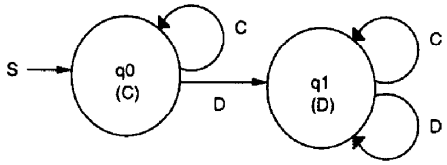
n	= 1	*fs*	= 0%
cr	= 1.0	*fc*	= 100%
dr	= 0.0		
ts	= 1.0		

All Defect



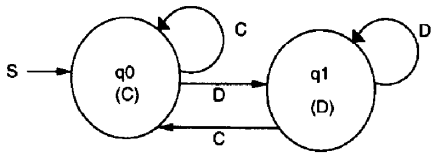
n	= 1	*fs*	= 0%
cr	= 0.0	*fc*	= 0%
dr	= 1.0		
ts	= 1.0		

Trigger



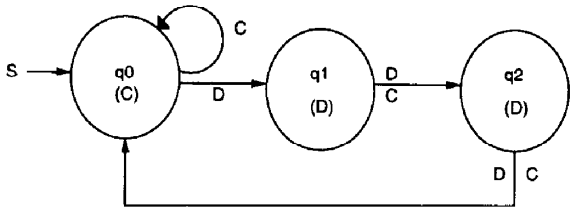
n	= 2	*fs*	= 10%
cr	= 0.5	*fc*	= 60%
dr	= 1.0		
ts	= 0.5		

Tit-For-Tat



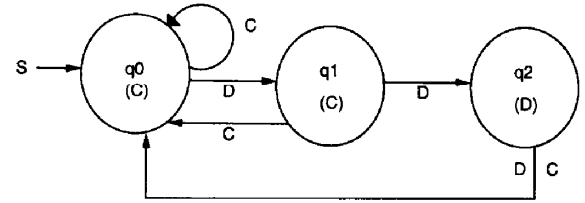
n	= 2	*fs*	= 10%
cr	= 1.0	*fc*	= 60%
dr	= 1.0		
ts	= 0		

Punish-Twice



n	= 3	*fs*	= 50%
cr	= 0.66	*fc*	= 60%
dr	= 0.66		
ts	= 0		

Forgive-Once, Punish-Once



n	= 3	*fs*	= 50%
cr	= 1.0	*fc*	= 60%
dr	= 0.33		
ts	= 0		

Fig. 3. Six different Moore machines.

A compact way to represent a Moore machine in the computer is to encode it as a string of binary bits. We illustrate our coding scheme with a Moore machine that can take up a maximum of sixteen states. Such a Moore machine can represent all strategies that base their moves on the full history of the previous two rounds (since each round of the game has four possible outcomes and two rounds of the game have sixteen possible outcomes). The string structure has a length of 144 bits consisting of sixteen nine-bit packets. In Fig. 4a, # is either 0 or 1. Note that the machine always start at state 0.

Each nine-bit packet represents an internal state of the machine. The first bit in a packet indicates the move of the machine whenever it enters that state, the next four bits give the next state if the opponent is observed to cooperate, and the last four bits give the next state if the opponent is observed to defect (see Fig. 4b).[6] For example, the Tit-For-Tat strategy is coded as follows:

0 0000 0001 1 0000 0001 # #### #### ··· # #### ####

This machine starts at state 0 and plays cooperation (bit 1). It does not switch the state if the opponent is observed to cooperate (bits 2–5). However, if the opponent defects, it switches to state 1 (bits 6–9). When the machine is in state 1, it plays defection (bit 10). If the opponent is observed to cooperate, it switches back to state 0 (bits 11–14). The machine does not switch the state if the opponent is observed to defect (bits 15–18). It is clear that the values of the remaining bits (19–144) are immaterial and would not affect the behavior of the machine. Any machine which has a size ranging from 1 to 16 can be represented in a similar manner.

Under the above coding scheme, any machine i that has a maximum of $n_{\max}$ states (we restrict it to be power of 2, i.e., $n_{\max} = 2^k$ where k is a positive integer) can be represented by a string of binary bits denoted by $B_i^1 B_i^2 \ldots\ldots B_i^{L-1} B_i^L$, where L is equal to $2^k(2k+1)$ (cf. Miller, 1989). See Holland (1975) for a detailed discussion on the advantages of the coding scheme.

For a sixteen-state machine, there are 2^{144} different possible structures under the above representation scheme. To find effective strategies in such a huge set, we need a very powerful technique. Holland's genetic algorithm is one such technique.

2.3. Genetic algorithm

The genetic algorithm works in five steps (see also Appendix I):

(1) An initial population of N players is chosen. Here, the individuals are 16-state Moore machines that are represented by 144-bit string structures. We

[6] A string of four bits can represent $2^4 = 16$ values.

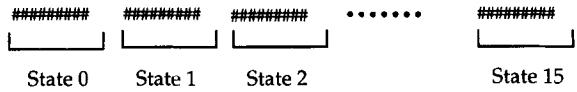


Fig. 4a. The coding scheme for a sixteen-state machine.

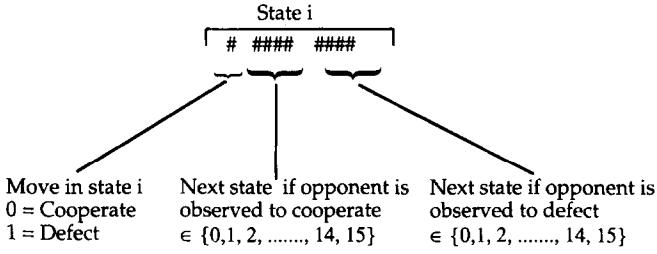


Fig. 4b. The coding of each nine-bit packet.

initialize all of the bits of the string structure except the bits that are allocated to moves (bits whose locations are $9xj + 1, j = 0,1,2, \ldots, 15$) by the outcomes from independent tosses of a fair coin. The bits that are allocated to moves (16 of them altogether) are initialized by the outcomes from tosses of a biased coin. Denote P_H as the probability of Head. Then, this initialization process has the effect of producing individuals that play a mixed strategy of $(1 - P_H)$ cooperation and P_H defection.[7] Thus, if we initialize all action bits by the outcome of a fair coin, the population is a group of individuals that will play cooperation and defection evenly in every round of the repeated game. The 'toughness' of the environment can be varied by changing the value of P_H. A population with individuals playing defection randomly 90% in every round of the repeated game can be initialized by the outcomes of a biased coin which has a P_H of 0.9. Initialize also the generation counter, g.

(2) Each individual (structure) is tested in the current environment to determine its fitness. Here this means that each individual plays a K-round RPD against all other individuals and a clone of himself/herself on a round robin basis. Individual i accumulates his/her payoff, $o(i,g)$, at generation g from playing the Prisoner's Dilemma $N \times K$ times. In games that incorporate implementation costs, we have two ways of charging our players. The first way is to charge an amount based on the size of machine. In every game that is played,

[7] Recall that the move of each Moore machine at state j is determined by the value of the bit at the location $9xj + 1$. We use the convention that a '1' (or Head) indicates a defection play and a '0' (or Tail) indicates a cooperation play.

a machine is charged an amount depending on its size, n, by one of the following equations:[8]

$$L(\beta, n) = \beta \times n \times R/16, \tag{1}$$

$$Q(\beta, n) = \beta \times n^2 \times R/16^2, \tag{2}$$

where β is the 'unit cognitive cost' that ranges from 0 to 1 and R is the reward (payoff) from mutual cooperation. For example, a ten-state machine has to pay a linear fee of $\beta \times 10 \times R/16$ or a quadratic fee of $\beta \times 100 \times R/256$ per game.

A linear cost function assumes that each machine state imposes a fixed memory burden on the player. It is reasonable if the total number of states to remember is small. If the total number of states to remember is large, each additional state might add to the effort of memorizing more than proportionately. A convex cost function may thus be more appropriate. We use a quadratic cost function to capture this 'increasing marginal effort' of remembering. The quadratic cost function charges a higher fee for each additional machine state in order to capture the increasing effort involved in memorizing each additional machine state.

Thus, the net accumulated payoff for individual i, $m(i, g)$, is given by

$$m(i, g) = o(i, g) - N \times K \times \{\text{Cost per period}\}. \tag{3}$$

The second way is to charge an amount based on the frequency of switching during the play of the repeated game. Denote the frequency of switching of machine i when it plays against j as $fs(i,j)$. The switching cost charged when i plays against j is given by the following equations:

$$L(\beta, fs(i,j)) = \beta \times K \times fs(i,j) \times R, \tag{4}$$

$$Q(\beta, fs(i,j)) = \beta \times K \times fs(i,j)^2 \times R. \tag{5}$$

Thus, a machine that switches more frequently pays a higher fee. The linear cost function means that each switching (or each retrieval of information from long-term to short-term memory) is equally expensive. The quadratic cost function penalizes more than proportionately on machines that switch states more frequently.

[8] The maximum size is 16. But a simple strategy (e.g., Tit-For-Tat) has a size (2) that is less than 16. A more complicated strategy will have a bigger size.

Here the net accumulated payoff for individual i, $m(i,g)$, is given by

$$m(i,g) = o(i,g) - \sum_{j=1}^{N} \{\text{Switching cost against } j\}. \tag{6}$$

We study how the introduction of a cognitive cost might impact the emergence of cooperative behavior. By varying β, we can study the impact of degree of 'bounded rationality' on emergence of cooperative behavior.

(3) Each individual's net accumulated payoff is normalized using the following transformation:

$$z(i,g) = \alpha + (m(i,g) - \mu(g))/\sigma(g), \tag{7}$$

where $\mu(g)$ is the population average and $\sigma(g)$ is the standard deviation of the net accumulated payoffs. The $z(i,g)$ is the fitness score of individual i at generation g. The z values which are below zero are truncated to zero. Notice that the transformation procedure is invariant to affine transformation. The parameter α determines the importance of relative performance.[9]

(4) Form a new population of N structures (for convenience, a constant population size is maintained):

(a) Reproduce N structures with $P(i) = z(i,g)/\sum_j z(j,g)$ and put them in the mating pool. An individual who has a higher fitness will have a larger representation in the mating pool.

(b) Randomly select two structures from the mating pool without replacement and form two children by using crossover operator with probability Pc to the structures. The crossover operator proceeds as follows: a single crossover point, $c \in \{1,2,\ldots, 142, 143\}$ (1 means crossover between 1 and 2, 2 means crossover between 2 and 3, etc.), is randomly selected on the bit string. The first child is formed by taking the first c bits from the first parent and attaching them to all of the bits after $c+1$ of the second parent. The second child is formed in a similar way using the remaining portions of the two parental strings. If we represent the parents by $B_a^1 B_a^2 \ldots B_a^{143} B_a^{144}$ and $B_b^1 B_b^2 \ldots B_b^{143} B_b^{144}$ then the new children are:

$$B_a^1 B_a^2 \ldots B_a^c B_b^{c+1} \ldots B_b^{143} B_b^{144},$$

$$B_b^1 B_b^2 \ldots B_b^c B_a^{c+1} \ldots B_a^{143} B_a^{144}.$$

[9] When α is infinity, the fitness scores are independent of net accumulated payoffs and every individual has an equal chance to mate. The choice of $\alpha = 2$ implies that individuals which are two standard deviations below the average are not allowed to mate (Miller, 1989).

(c) Mutate the newly formed children with probability $P_m(g) = P_m(0)e^{-0.693g/\zeta}$ where ζ is the half life of the exponential decay.[10] We choose an exponentially decaying mutation rate because we expect players to experiment less frequently as they 'progress through' the generations (see below for discussions on the psychological interpretations of the genetic operators). The mutation operator is performed on a bit-by-bit basis. Mutation occurs when a bit changes states. A natural way to represent the mutation rate is by the average number of mutations per string structure. Here, an initial mutation rate of two bits per string structure ($P_m(0)$) is equivalent to a mutation rate of $2/144 = 1.4\%$.

(d) Repeat (b) to (c) until N new structures are formed.

(5) The new population will exhibit patterns of behavior that are more like those of the successful individuals of the previous generation, and less like those of the unsuccessful ones. With each new generation, the individuals with relatively high fitness will be more likely to pass on parts of their strategies, while the relatively unsuccessful individuals will be less likely to have any parts of their strategies pass on. Increment g by 1 and go to step 2 (next generation).

The reproduction, crossover, and mutation operators have appealing psychological interpretations. We can interpret the reproduction operator as either intelligent learning by imitation, or simply blind imitation of the successful players. The crossover operator introduces innovation into the evolutionary process. The operator combines or mixes two successful strategies (parents in the mating pool) into two possibly even more successful strategies (the newly formed children). The mutation operator is similar to 'trial and error' learning. It represents a random experimentation process. Both the crossover and mutation operators introduce variety into the evolutionary process. However, there is a difference between the two: the crossover operator represents a discovery mechanism that entails the juxtaposition of different strategies, and the mutation operator represents a random walk that avoids 'trapping' into suboptimal solution. Thus, a new strategy is either constructed from the successful strategies of previous generations, or it is a random variant of an old strategy.

3. Simulation results

3.1. Design of experiments

We ran three simulation experiments. The first experiment was designed to test the robustness of the evolution of cooperation in hostile environments. The

[10] For example, if ζ is 10, $P_m(g)$ drops by half after every ten generations.

degree of hostility of the environment was controlled by varying P_H. Six values of P_H were used: 0.5 to 1.0, with an incremental step of 0.1. When P_H is 1.0, we have an environment in which all players play defect in every around of the repeated game.

The second and third experiments were designed to study the effects of implementation costs on the emergence of cooperative behavior. Each experiment had two sets of simulation runs: one set used a linear cost function and the other used a quadratic cost function. In the second experiment, players were charged a fee based on the size of machine; in the third experiment, they were charged a fee based on the frequency of switching. In all four sets of simulation runs, the proportional factor, β, was varied from 0.05 to 0.25, with an incremental step of 0.05. A β of 0.05 would mean that a sixteen-state machine is charged a fee of 5% of the Pareto-optimal payoff (i.e., R) per game.

All simulation runs had 100 generations. Simulation results show that doubling the number of generation would only increase the payoff marginally with the choice of parameters given below. The settings of the parameters are summarized in Table 1. A few remarks about the choices of parameters are appropriate here.

Table 1
The settings of parameters in the simulation experiments

Prisoner's Dilemma game
$T = 5$ $R = 3$ $P = 1$ $S = 0$
Game is repeated 150 rounds, i.e., $K = 150$
Automata (strategy)
A maximum of 16 states, i.e., $n_{max} = 16$
Length of the string, L, is 144
Population
Population size $N = 30$
P_H is a variable in Experiment I and is set to 0.5 in Experiments II and III
Genetic algorithm
$P_c = 0.6$ $P_m(0) = 4$ bits per string structure $\zeta = 10$
Normalization parameter $\alpha = 2$
Costs
β is set to 0 in Experiment I and is a variable in Experiments II and III

First, the parameters of the Prisoner's Dilemma game were standard (i.e., same as the parameters used in Axelrod, 1984, 1987; Miller, 1989). These parameters were chosen such that they would provide a sufficient level of conflict of interest (see Rapoport, 1966, for a measure of conflict of interest). Simulation results show that the patterns of results reported below are not sensitive to changes in these parameters. The game was chosen to repeat 150 rounds so that each individual could differentiate one from another. A lower number of rounds, e.g., 50–100, would slow down the evolution process but would not affect the patterns of results.

Second, we limit the size of machine to 16 states because it could capture relatively complex RPD strategies without placing too much burden on the computing resources. Simulation results show that this constraint on the size of machine is not binding, i.e., successful automata have sizes less than 16 states.

Third, the size of the population was chosen to be 30. Simulation results show that a larger population would not affect the emergence of behaviors, but would increase the time of simulation substantially. On the other hand, a smaller population would not provide enough diversity in strategies for the genetic algorithm to perform well. The value of P_H was a variable in Experiment I. In the second and third experiments, it was fixed at 0.5.

Last, we chose P_c, $P_m(0)$, β, ζ, and α as follows. The values for P_c and $P_m(0)$ were set to 0.6 and 4 bits per string structure, respectively. Similar values were also used in previous studies (see Axelrod, 1987; Miller, 1989). β was set to zero in Experiment I. In Experiments II and III, it varied from 0.05 to 0.25. ζ was chosen to be 10 by trial and error. α was set to 2 in all three experiments because we did not want individuals who were two standard deviations below average to mate. Interestingly, simulation results show that there are large equivalent classes of parameters that would yield cooperative behavior (see Appendix II). In Section 3.5, we conduct sensitivity analysis of the results with respect to α, β, and ζ.

In all the results presented below, all variables are averages over 100 simulation runs and thirty members of each population conditional on the generation.

3.2. *Experiment I: Varying the toughness of the initial environment*

In Experiment I, P_c, $P_m(0)$, and ζ were fixed at 0.6, 4 bits per string structure, and 10 respectively. P_H varied from 0.5 to 1.0. The simulation results for the case where $P_H = 0.5$ replicate Miller's (1989) results. The simulation results for other P_H values suggest that emergence of cooperative behavior is not sensitive to changes in the nature of the initial environment. In addition, they show that cooperation can be started by a small number of individuals or mutants who are prepared to cooperate, even in a world where no one else will do so.

Figs. 5a–g plot the variations of the variables over 100 generations for $P_H = 0.5, 0.7, 1.0$. The average frequencies of cooperation are 86%, 88%, and

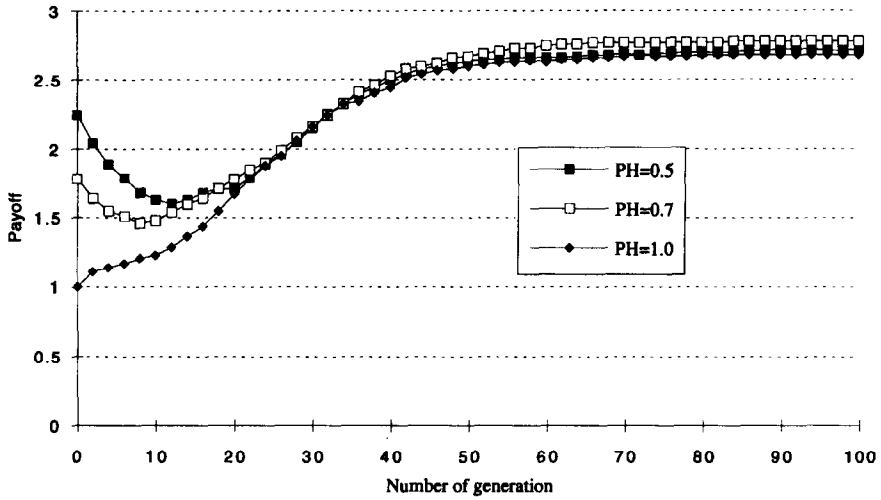


Fig. 5a. Payoff versus number of generation.

83%, respectively. The variances of the frequencies of cooperation are 0.0006, 0.0005, and 0.0006.[11]

Figs. 5a–b show that the emergence of cooperative behavior is not sensitive to changes in the nature of the initial environment. Making the initial environment tougher does not stop individuals in the population from developing cooperative behavior.[12] This finding implies that as long as individuals are willing to experiment and employ learning rules that have evolutionary properties, cooperative behavior in a finitely RPD can emerge. Notice that for $P_H = 0.5$ and 0.7, the population tends to defect in the early generations (this pattern also occurs when $P_H = 0.6$ and 0.8). This phenomenon does not, however, continue; the population begins to cooperate after about 10 generations. Notice that the initial environment has strategies that are randomly generated. These random strategies do not have predictable patterns, and a good strategy in such an environment is to defect more frequently. As the strategies become more predictable, players quickly learn that it is not in everyone's interest to defect, and they begin to cooperate.

An analysis of the evolved machines indicates that they tend to exhibit behavior that is quite similar to Tit-For-Tat. A typical successful strategy starts

[11] The variance increases and peaks at around 28th generation. It then decreases steadily when emergence of behavior occurs. This pattern occurs in all three cases.

[12] Bear in mind that the parameters of the genetic algorithm were fixed at values that were 'tuned' for the case where $P_H = 0.5$.

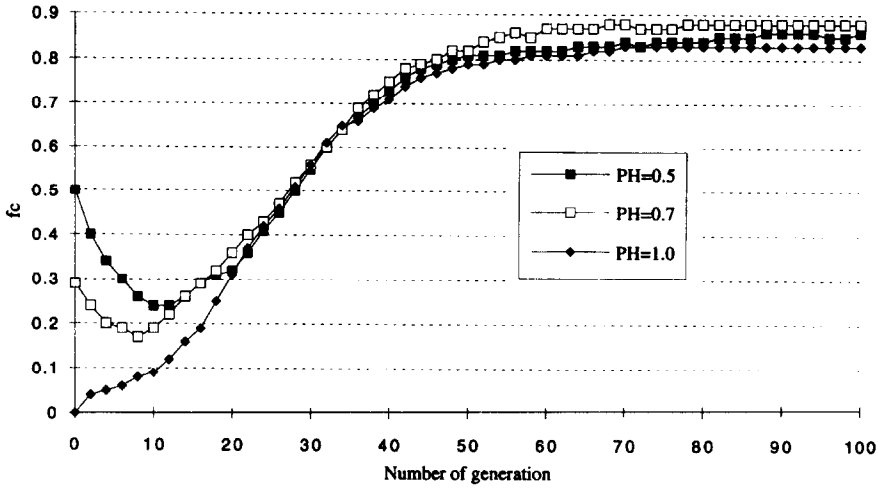


Fig. 5b. *fc* versus number of generation.

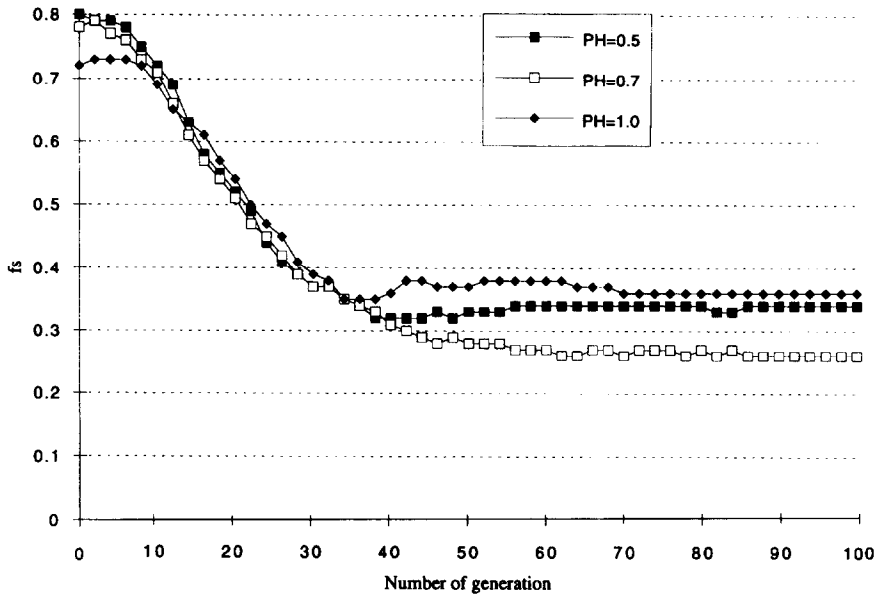


Fig. 5c. *fs* versus number of generation.

by cooperating and continues to do so as long as the opponent cooperates. The strategy differs from Tit-For-Tat in the way it punishes defection. It tends to enter into a 'sophisticated way of defending'. For example, immediately after a defection by the opponent, an evolved machine may monitor the moves by the

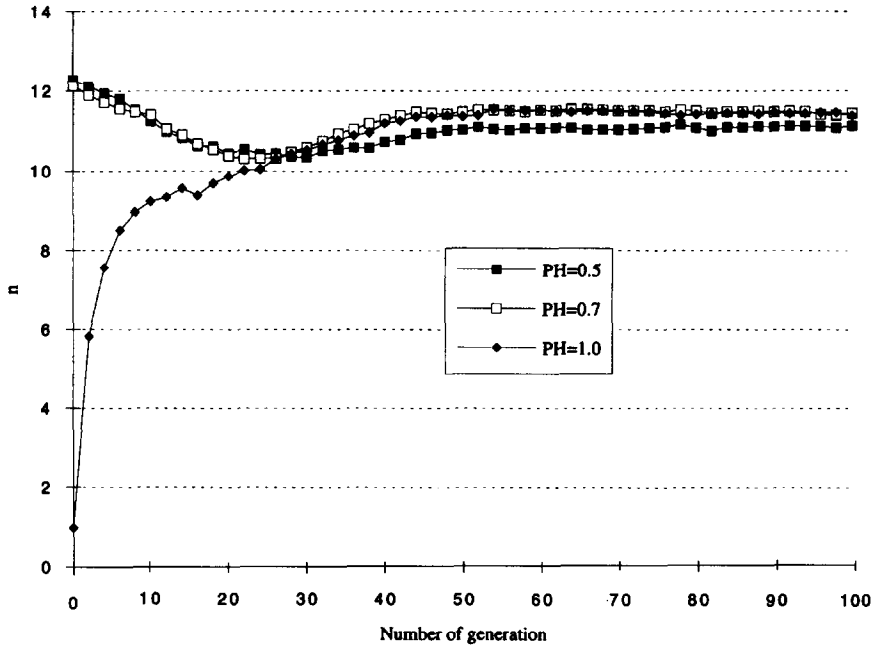


Fig. 5d. n versus number of generation.

opponent and only return to the initial cooperative state if the opponent defects not more than once in three periods.

Fig. 5c shows that individuals learn not to switch their internal states too frequently in later generations (the average frequency of switching starts from 0.8 and drops to 0.32 at the 100th generation). A high level of switching can jeopardize the coordination of moves and hence lower the frequency of cooperation. In addition, it makes players' moves less predictable. This result suggests that it is harmful for individuals to switch states too frequently in playing an RPD.

Fig. 5d shows that the size of a machine is not a binding constraint in playing an RPD game. Successful strategies in a finitely RPD in a costless environment need not be too complex. When $P_H = 1.0$, the size of a machine starts from 1 because we use the size of minimal machine. Notice that all machines which have more than one accessible state, but always defect, are equivalent in behavior to a one-state machine (minimal machine) that always defects, regardless of the opponents' moves. Overall, the results suggest that thriving strategies have a size of 11.[13]

[13] The averages are 11.12 ($P_H = 0.5$), 11.44 ($P_H = 0.7$), and 11.38 ($P_H = 1.0$). The variances are 0.037, 0.016, and 0.014.

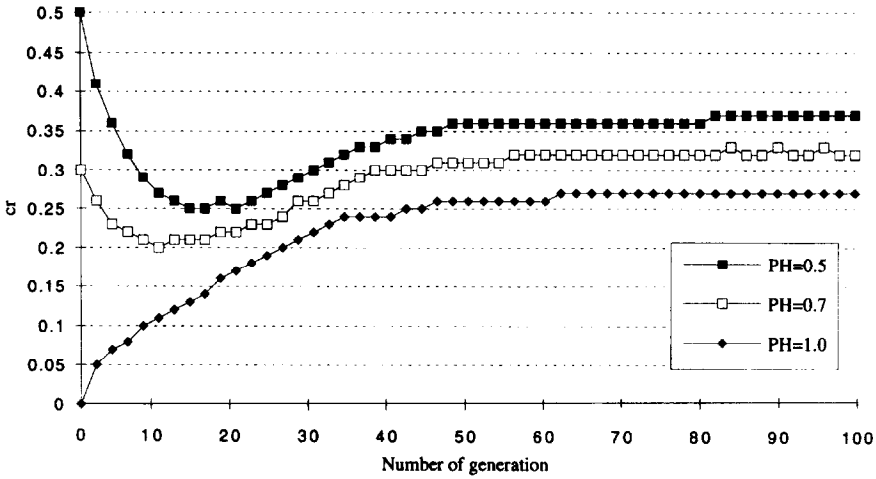


Fig. 5e. *cr* versus number of generation.

In Figs. 5e–f, we see that evolved strategies have much higher defection than cooperation reciprocities.[14] Note that Tit-For-Tat has equal defection and cooperation reciprocities. Thus the thriving strategies are more 'punishing' than Tit-For-Tat and can better exploit 'nice' strategies. It is worthwhile noting that the defection reciprocity increases and the cooperation reciprocity decreases with the hostility of the environment (P_H).

Fig. 5g shows that successful strategies have very few terminal states (*ts* has an average of 0.030 and a variance of 0.0001 at the 100th generation). Since most terminal states exhibit defection, this finding suggests that successful strategies in a costless environment are not simple trigger strategies.

3.3. *Experiment II: Implementation costs measured by size of machine*

In Experiment II, P_H, P_c, $P_m(0)$, and ζ were fixed at 0.5, 0.6, 4 bits per string structure, and 10, respectively. Figs. 6a–c and 6d–f graph the variations of *fc*, *n*, and *ts* for $\beta = 0.05$, 0.15, and 0.25 for the linear and quadratic cost functions, respectively. Overall, the simulation results show that introducing a small implementation cost measured by the size of the machine destroys development of cooperative behavior.

Figs. 6a and 6d show that the level of cooperation drops substantially. For instance, the level of cooperation for the case $L(0.25,n)$ drops to as low as 15%.

[14] The average defection and cooperation reciprocities are 0.70 and 0.37 ($P_H = 0.5$), 0.74 and 0.32 ($P_H = 0.70$), and 0.76 and 0.27 ($P_H = 1.0$). The variances are all below 0.0001.

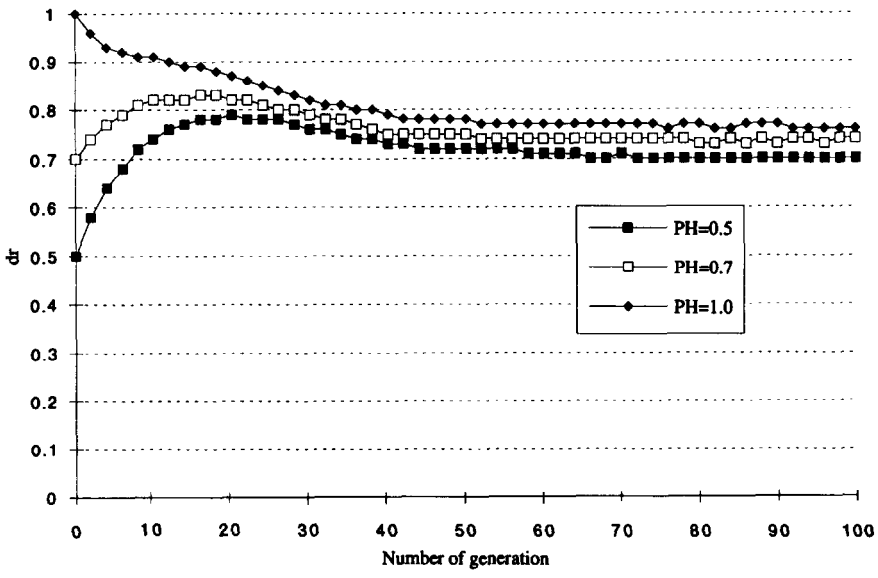


Fig. 5f. *dr* versus number of generation.

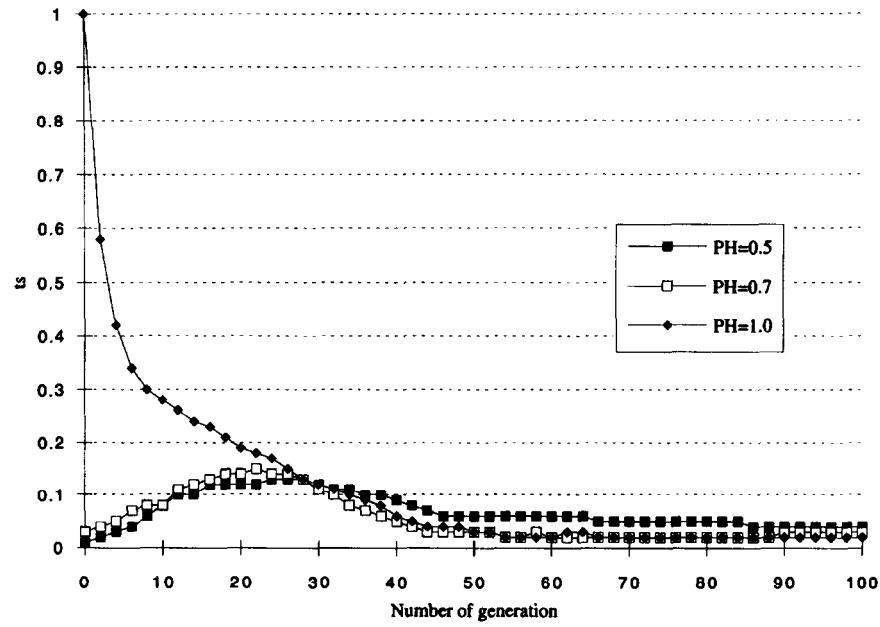


Fig. 5g. *ts* versus number of generation.

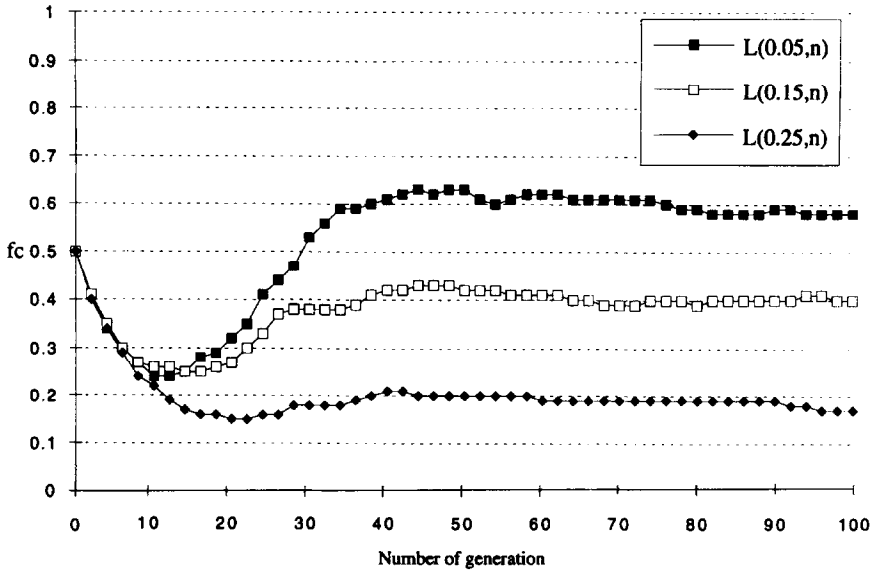


Fig. 6a. *fc* versus number of generation.

The variances of the frequency of cooperation in all cases are below 0.0023. Thus, the simulation results suggest that cooperative behavior can be destroyed when one moves into a costly world.[15]

Figs. 6b and 6e suggest that the sizes of the thriving strategies range from 1.3 to 3.1 depending on the value of β. When β is large (e.g., 0.20 and 0.25), the evolved strategies have sizes ranging from 1 to 2. An analysis of the string structures over the simulation runs suggest that a majority of these strategies are 'All *D*'. When β is small (0.05 and 0.10), the sizes of the machines range from 1 to 4 and the type of strategies is more diverse. It includes 'All *D*', Trigger, and some hostile strategies that begin the game by playing defection. Figs. 6c and 6f indicate that the evolved strategies have a high proportion of terminal states. Since most machines play defection in their terminal states, there is a high percentage of individuals who use the rule 'never trust your opponent again'. The use of this kind of rule reduces the size of the machine, but at the expense of obtaining a higher payoff from mutual cooperation.

[15] The frequencies of cooperation are significantly different from the frequency of cooperation for $\beta = 0$. When a linear cost is charged, the *t*-statistics are 5.2 ($\beta = 0.05$), 8.5 ($\beta = 0.15$), and 12.8 ($\beta = 0.25$). When a quadratic cost is charged, the *t*-statistics are 4.6 ($\beta = 0.05$), 8.3 ($\beta = 0.15$), and 9.3 ($\beta = 0.25$). All are significant at the 1% level.

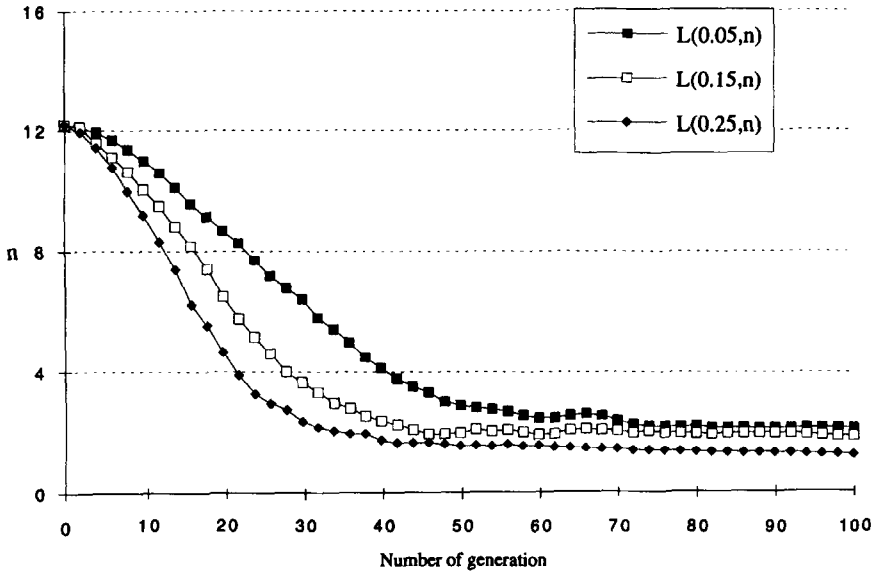


Fig. 6b. n versus number of generation.

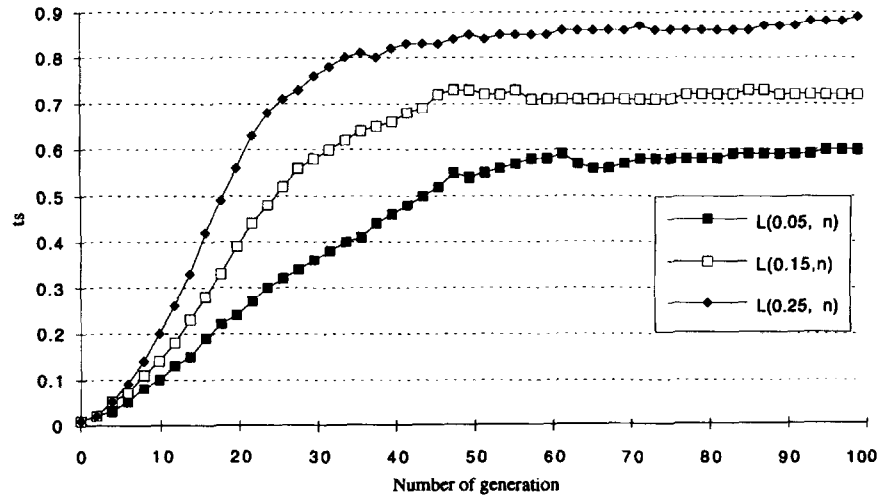


Fig. 6c. ts versus number of generation.

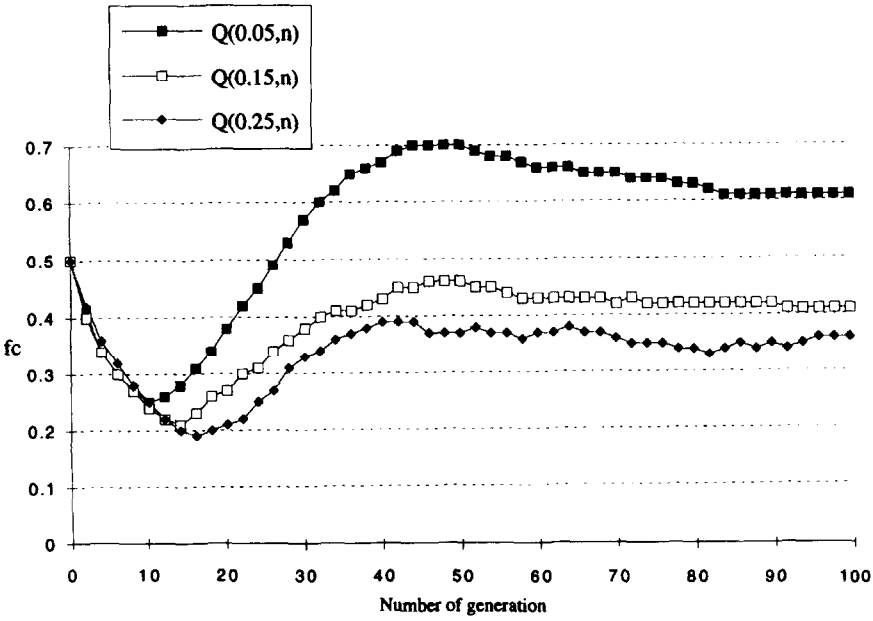


Fig. 6d. *fc* versus number of generation.

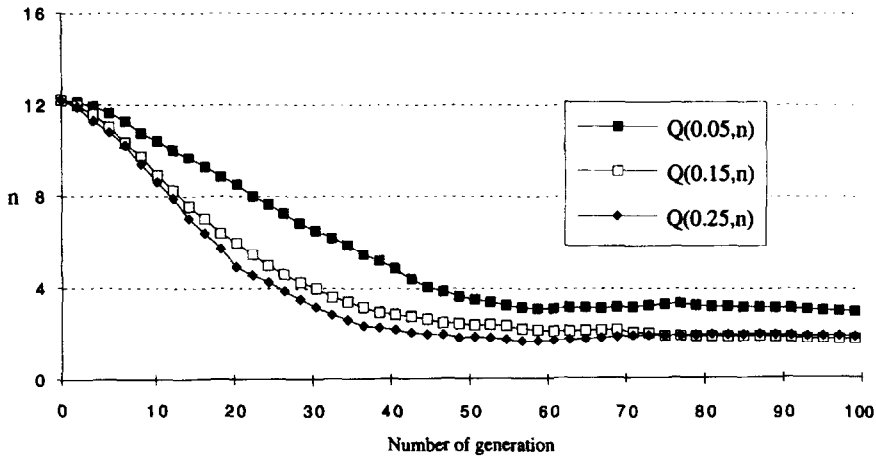


Fig. 6e. *n* versus number of generation.

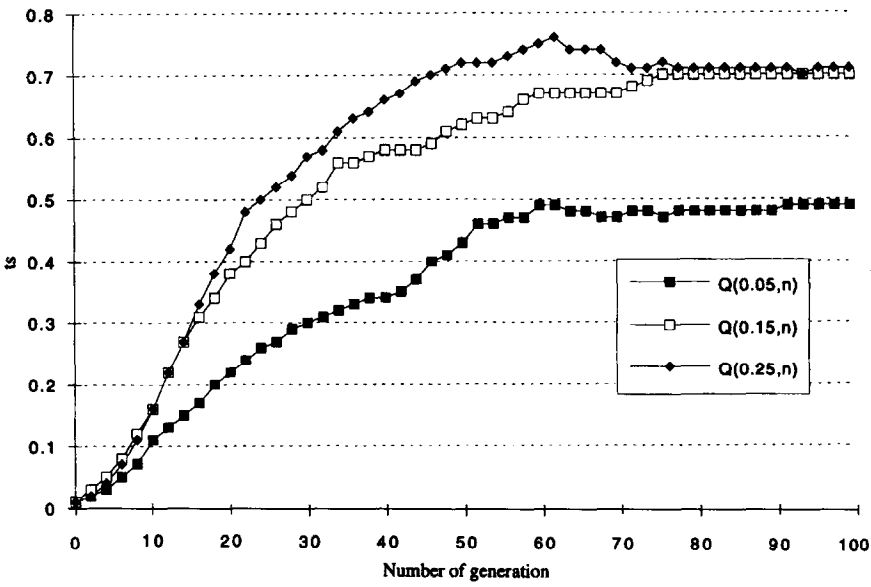


Fig. 6f. *ts* versus number of generation.

3.4. *Experiment III: Implementation costs measured by frequency of switching*

In Experiment III, P_H P_c, $P_m(0)$, and ζ were fixed at 0.5, 0.6, 4 bits per string structure, and 10, respectively. Figs. 7a–b and 7c–d graph the variations of *fc*, *fs*, *ts* for $\beta = 0.05, 0.15, 0.25$ for the linear and quadratic cost functions, respectively. Overall, the results suggest that penalizing a complex strategy based on frequency of switching will not hurt the development of cooperative behavior.

Fig. 7a shows that charging a fee which varies linearly with the frequency of switching will not affect the development of cooperation. The levels of cooperation at the 100th generation are 90%, 91%, and 85%. The variances of the frequency of cooperation over the 100 simulation runs are below 0.0012.

Charging a fee that varies in a quadratic manner with the frequency of switching may help cooperation if it is charged at an optimal level (see Fig. 7d). When β is 0.25, the level of cooperation achieved is as high as 96%.[16] In other cases, it is at least 90%. The variance of the frequency of cooperation in all cases is below 0.0005.

Figs. 7b and 7e show that the ideal level of switching is quite low in an RPD game. Thriving strategies do not switch their states too frequently because if

[16] When β was increased to 0.30, the level of cooperation dropped to about 92%. A *t*-test shows that the improvement in frequency of cooperation is significant (*t*-statistic = 2.8, $p < 0.01$).

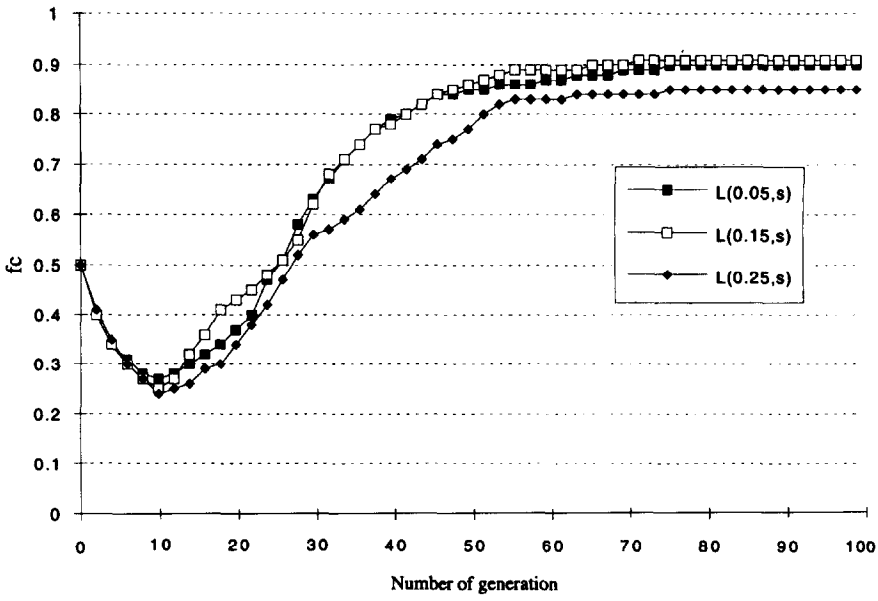


Fig. 7a. *fc* versus number of generation.

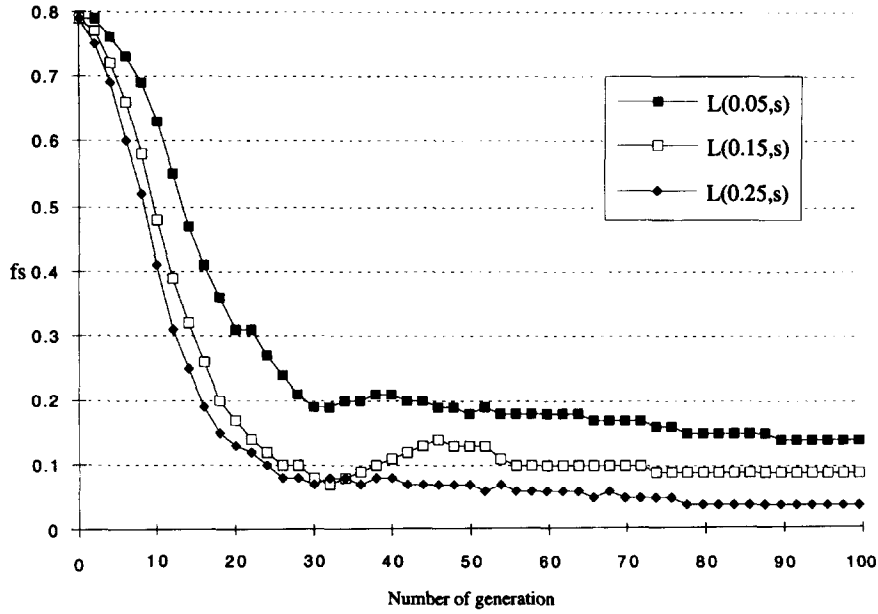


Fig. 7b. *fs* versus number of generation.

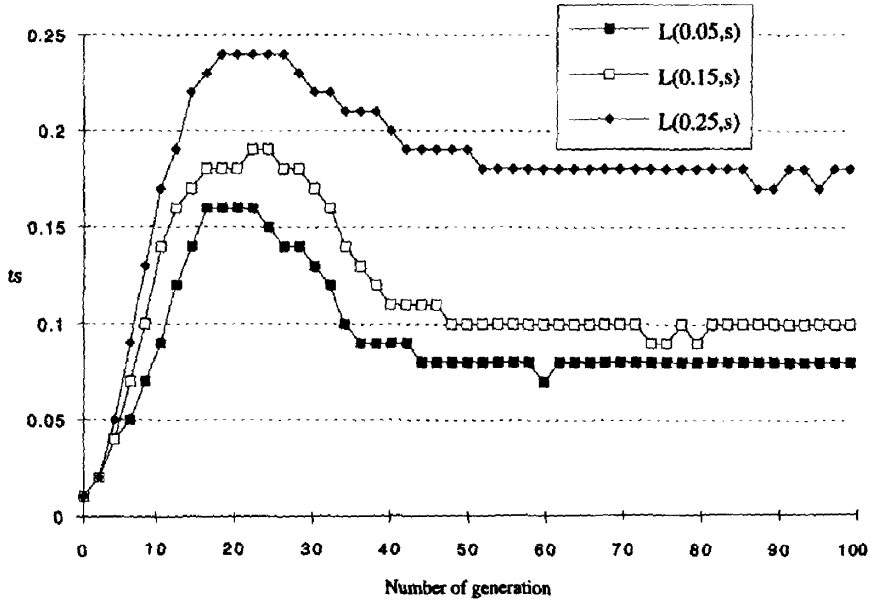


Fig. 7c. *ts* versus number of generation.

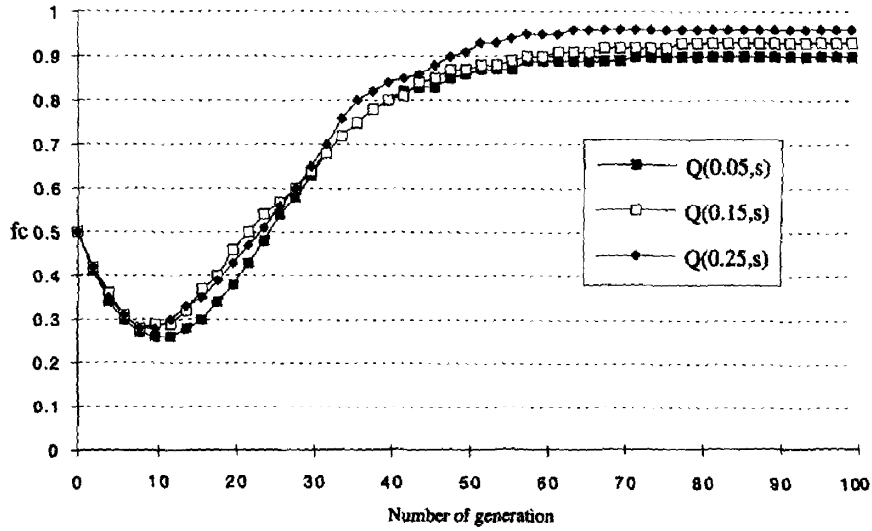


Fig. 7d. *fc* versus number of generation.

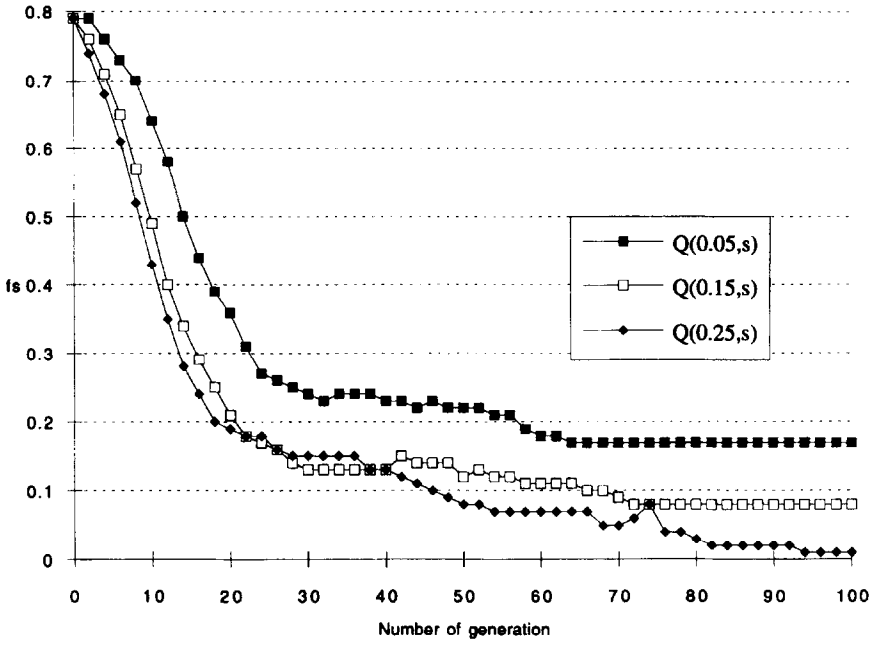


Fig. 7e. *fs* versus number of generation.

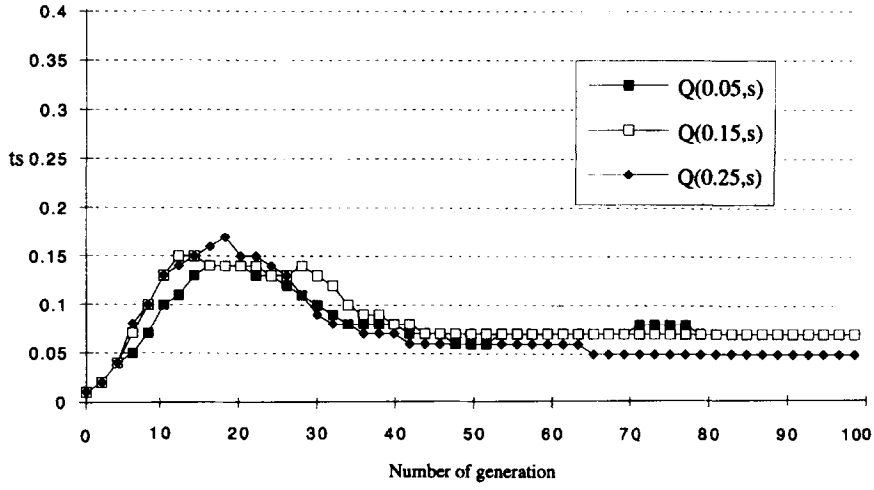


Fig. 7f. *ts* versus number of generation.

they do they run the risk of losing coordination in their moves. They switch states only if their opponents change their 'intentions'. An analysis of the string structures reveals that most individuals switch to play defection after detecting a defection from their opponents.

3.5. *Sensitivity analyses*

The simulation results we obtain might be sensitive to the values of some of the parameters we chose. We test the sensitivity of our findings with respect to: the importance of relative performance, α, the unit cognitive cost, β, and the half-life of mutation, ξ. We repeat all three experiments for β from 0.005 to 0.04 in steps of 0.005, for three values of α (1.0, 1.5, 2.5) and for two values of ξ (25, 50). For $\beta < 0.035$, cooperation can still emerge in Experiment II. Thus, our finding from Experiment II is robust to changes in β only if it is above 0.035. The findings are robust to changes in α. In all three experiments, the level of cooperation changes less than 5% when α was varied. A large ξ appears to slow down the emergence of behavior but does not change its pattern. For instance, a slower decay of mutation slows down but never prevents the emergence of cooperative behavior in Experiments I and III. In Experiment II, cooperative behavior stills breaks down with a slower rate of mutation.[17]

4. Discussion

In summary, our simulation results show:

(i) The evolution of cooperation is not sensitive to the hostility of the initial environment. This finding implies that as long as individuals are willing to experiment and employ learning rules that have evolutionary properties, cooperative behavior in a finitely RPD can emerge. Cooperation can be started by a small number of individuals or mutants who are prepared to cooperate, even in a world where no one else will do so. These mutants thrive by obtaining a higher payoff when they play cooperation against each other. The defectors die off because they 'kill' each other and have performance worse than that of the cooperators.

Our successful strategies are neither Tit-For-Tat nor Trigger. They are more complex and have more than ten states. These strategies tend to reciprocate cooperation less frequently. This behavior has been observed by some psychologists. For example, Ross and Sicoly (1979) observed that people tend to remember their own 'niceness' more than others' 'niceness' because of egocentric bias. Consequently, they may not reciprocate others 'niceness' as readily as they expect others to reciprocate their own 'niceness'.

[17] The details of the simulation results can be obtained from the author.

The evolved strategies also punish defection differently. Tit-For-Tat forgives their opponents immediately after a cooperative move and Trigger never forgives their opponents. Our successful strategies are more 'suspicious' than Tit-For-Tat as they monitor the opponents' moves immediately after a defection for a few rounds. If the opponents are nice, they reciprocate by playing cooperation. Unlike Trigger, these strategies will return to cooperative behavior as long as the opponents are subsequently 'detected' to be 'nice'. Such a feature can be very useful in a noisy environment where moves of opponents may be misrepresented. Thus the evolved strategies are neither as 'naive' as Tit-For-Tat nor as 'rigid' as Trigger.

The evolved strategies have higher defection than cooperation reciprocities (Tit-For-Tat has equal defection and cooperation reciprocities). In addition, the defection reciprocity increases and cooperation reciprocity decreases with P_H. This is interesting. It implies that evolved strategies from a more hostile environment are more ready to punish than forgive. Cultures that tend to exhibit such 'reciprocity bias' might be due to hostility in their historical environments.

(ii) Penalizing a complex strategy based on the size of the machine can destroy players' cooperative behavior. Our simulation results show that if a sixteen-state machine is charged a fee of 5% of the Pareto-optimal payoff (i.e., R) per game, the level of cooperation can drop substantially (from 86% to 58%). While it is clear that penalizing a complex strategy will favor a simple strategy, it is not obvious that it will destroy the emergence of cooperative behavior. This finding suggests that we may not observe cooperative behavior in those real-life settings where the costs of monitoring are high or the process of monitoring is impossible. Examples include arms race and trade conflicts.

Combined with the results from Experiment I, these results suggest that, if it is costly to monitor the opponents' moves, it is better playing a simple and unforgiving strategy. An unforgiving strategy can better exploit irrational strategies introduced by the mutation process than a forgiving one (Linster, 1992). Linster shows that the ability of a strategy to exploit poor strategies may determine its degree of success in an evolutionary contest. A forgiving strategy like Tit-For-Tat, while simple, is not as good in exploiting irrational strategies. This may explain why our successful strategies are unforgiving. In real life, unforgiving strategies seem pragmatic because they tend to require less monitoring.

Linster (1992) shows that if the complexity cost is not too large, the proportions of the population can enter a cycle. We did not observe such a cyclic pattern of behavior in our simulation runs. This may be due to three reasons. First, we allow a much larger strategy space (a maximum of sixteen-state instead of two-state machines). Second, our evolutionary process includes the crossover operation and Linster's does not. Third, we use a decaying mutation rate and Linster uses a constant mutation rate over time. Thus Linster's populations are repeatedly perturbed over generations while players in our populations experiment less frequently as they 'progress through' the generations.

Our sensitivity analysis shows that cooperative behavior can still emerge if $\beta < 0.0035$. This finding suggests that one way to promote cooperative behavior is to make the monitoring of strategies as simple as possible. This can be accomplished by chunking of information or applying invariant game transformations (see Ho and Weigelt, 1993). Thus, as the need for functional integration heightens in organizations, the costs of monitoring should become an important factor in organizational design.

(iii) A linear cost measured by frequency of switching will not hurt cooperation. Simulation results from Experiment I show that successful strategies in a costless environment switch states in about 30% of the moves (see Fig. 5c). When we charge a fee that increases linearly with the frequency of switching, it drops to about 10%. Since the evolved strategies have similar number of states (about 10–11 states), this implies that the evolved strategies are less sophisticated than those picked in the costless environment. An analysis of the string structures shows that these strategies tend to have more terminal states and they reciprocate both defection and cooperation more readily than the evolved machines in the costless environment.

A quadratic cost measured by frequency of switching can help cooperation if the fee is charged at a suitable level. This result is interesting. Upon examining the evolved strategies, we find that the successful strategies are different from those of the linear case; they have fewer terminal states (see Figs. 7c and 7f). Thus the evolved strategies are more flexible and forgiving. While Banks and Sudaram (1990) show that costs of switching can destroy cooperation, our results reveal that this finding may depend on the way the switching cost is charged.

In all three experiments, the number of terminal states and the level of cooperation correlate well. A lower number of terminal states is always accompanied by a higher level of cooperation. This is because 95% of the terminal states play defection. This finding suggests that maintaining 'flexibility' is important for cooperative behavior. A machine that has a lot of terminal states is rigid because it cannot react to changes in opponents' behaviors.

Appendix I: Genetic algorithm software

The genetic algorithm software is written in Pascal by the author. The software runs on the VAX 6400 super minicomputer. In addition to what has been described in Section 2, we make the following modifications to make the algorithm more robust:

(i) Use a stochastic remainder selection scheme rather than a basic selection scheme to select individuals into the mating pool. For a constant population of

N individuals, the basic scheme performs N Bernoulli trials, and selects individual i with a probability $P(i) = \mu(i,g)/\sum_j \mu(j,g)$ in each trial. The stochastic remainder selection scheme works in a slightly different manner. First we compute the expected number of individual i as $e_i = N \times \mu(i,g)/\sum_j \mu(j,g)$. Then, each individual i is allocated samples according to the integer part of the e_i values. Finally, the fractional parts of the expected number values are treated as probabilities and are used to fill the remaining population slots until the whole population is filled. Notice that the new scheme yields the same expected number of individuals, but with a smaller variance. It has been shown that the stochastic remainder selection scheme tends to improve the performance of the genetic algorithm (Goldberg, 1989).

(ii) Use two crossover points instead of one. If we treat a string structure as a ring with no beginning or end, i.e., with the first bit immediately following the last bit, then it becomes clear that there are in fact two crossover points: one fixed at position 0 and the other randomly selected. An immediate generalization to this basic crossover operator is to allow both crossover points to be randomly selected. It has been shown that a two-point crossover operator tends to lead to better performance compared to a one-point crossover operator.

Appendix II

Table 2 shows twelve different sets of parameters and their associated payoffs and frequencies of cooperation at the 100th generation. (P_H is set to 0.5 in all these simulations.) The table illustrates that there are large equivalent classes of parameters that would yield emergence of cooperation behavior.

Table 2
Payoffs and frequencies of cooperation at 100th generation produced by different sets of genetic algorithm parameters

P_c	$P_m(0)$	ζ	Payoff(100)	$fc(100)$
0.54	2	25	2.75	0.87
0.45	2	20	2.72	0.84
0.45	4	10	2.80	0.89
0.50	2	25	2.79	0.88
0.50	2	20	2.76	0.87
0.50	4	10	2.80	0.89
0.55	2	25	2.67	0.82
0.55	2	20	2.75	0.87
0.55	4	10	2.78	0.88
0.60	2	25	2.76	0.87
0.60	2	20	2.65	0.81
0.60	4	10	2.71	0.86

References

Abreu, D. and A. Rubinstein, 1988, The structure of Nash equilibrium in repeated games with finite automata, Econometrica 56, 1259–1281.

Anderlini, L., 1989, Communication, computability, and common interest games, Working paper (Santa Fé Institute, Santa Fé, NM).

Aumann, R., 1981, Survey of repeated games, in: R.J. Aumann et al., eds., Essays in game theory and mathematical economics in honor of Oscar Morgenstern, 11–42.

Aumann, R., 1989, Perspectives on bounded rationality, Working paper (Stanford Graduate School of Business, Stanford, CA).

Axelrod, R., 1984, The evolution of cooperation (Basic Books, New York, NY).

Axelrod, R., 1987, The evolution of strategies in the iterated prisoner's dilemma, in: Lawrence Davis, ed., Genetic algorithms and simulated annealing (Morgan Kaufman, Los Altos, CA).

Banks, J. and R. Sundaram, 1990, Repeated games, finite automata, and complexity, Games and Economic Behavior, 97–117.

Binmore, K., 1987, Modeling rational players: Part I, Economics and Philosophy 3, 179–214.

Binmore, K., 1988, Modeling rational players: Part II, Economics and Philosophy 4, 9–55.

Binmore, K. and L. Samuelson, 1990, Evolutionary stability in repeated games played by finite automata, Working paper (University of Wisconsin, Madison, WI).

Brown, G.W., 1951, Iterative solution of games by fictitious play, in: Activity analysis of production and allocation (Wiley, New York, NY).

Friedman, D., 1991, Evolutionary games in economics, Econometrica 59, 637–666.

Fudenberg, D. and D. Kreps, 1988, A theory of learning, experimentation, and equilibrium in games, Memo (Stanford Graduate School of Business, Stanford, CA).

Fudenberg, D. and E. Maskin, 1986, The Folk theorem in repeated games with discounting or with incomplete information, Econometrica 54, 533–554.

Fudenberg, D. and E. Maskin, 1990, Evolution and cooperation in noisy repeated games, American Economic Review 80, 274–279.

Gilboa, I., 1988, The complexity of computing best-response automata in repeated games, Journal of Economic Theory 45, 342–352.

Goldberg, D., 1989, Genetic algorithms in search, optimization and machine learning (Addison-Wesley, New York, NY).

Harrison, M., 1965, Introduction to switching and automata theory (McGraw-Hill, New York, NY).

Ho, T.-H. and K. Weigelt, 1993, Task complexity, equilibrium selection, and learning: An experimental study, Management Science, forthcoming.

Holland, J.H., 1975, Adaptation in natural and artificial systems (University of Michigan Press, Ann Arbor, MI).

Holland, J.H. and J. Miller, 1991, Artificially adaptive agents in economic theory, American Economic Review 81, 365–370.

Holland, J.H., K.J. Holyoak, R.E. Nisbett, and P.R. Thagard, 1986, Induction: Processes of inference, learning, and discovery (MIT Press, Cambridge, MA).

Hopcroft, J. and J. Ullman, 1979, Introduction to automata theory, languages, and computation (Addison-Wesley, Reading, MA).

Kalai, E. and W. Stanford, 1988, Finite rationality and interpersonal complexity in repeated games, Econometrica 56, 397–410.

Kreps, D., 1990, Game theory and economic modeling (Oxford University Press, Oxford).

Kreps, D., P. Milgrom, J. Roberts, and R. Wilson, 1982, Rational cooperation in the finitely repeated prisoner's dilemma, Journal of Economic Theory 27, 245–252.

Linster, B., 1992, Evolutionary stability in the infinitely repeated prisoners' dilemma played by two-state Moore machines, Southern Economic Journal, 880–903.

Marimon, R. and J. Miller, 1989, Money as a medium of exchange in an economy with genetically reproduced decision rules, Working paper (Santa Fé Institute, Santa Fé, NM).
Marimon, R., E. McGrattan, and T. Sargent, 1990, Money as a medium of exchange in an economy with artificial intelligent agents, Journal of Economic Dynamics and Control 14, 329–374.
Milgrom, P. and J. Roberts, 1991, Adaptive and sophisticated learning in repeated normal form games, Games and Economic Behavior 3, 82–100.
Miller, J., 1989, The coevolution of automata in the repeated prisoner's dilemma, Working paper (Santa Fé Institute, Santa Fé, NM).
Minsky, M., 1967, Computation: Finite and infinite machines (Prentice-Hall, Englewood Cliffs, NJ).
Neyman, A., 1985, Bounded rationality justifies cooperation in the finitely repeated prisoner's dilemma, Economics Letters 19, 227–229.
Radner, R., 1986, Can bounded rationality resolve the prisoner's dilemma?, in: Essays in honor of Gerard Debreu (North-Holland, Amsterdam).
Rapoport, A., 1966, A note on the index of cooperation for prisoner's dilemma, Journal of Conflict Resolution, 100–103.
Rapoport, A. and A.M. Chammah, 1965, Prisoner's dilemma (University of Michigan Press, Ann Arbor, MI).
Ross, M. and F. Sicoly, 1979, Egocentric biases in availability and attribution, Journal of Personality and Social Psychology 37, 322–336.
Rubinstein, A., 1986, Finite automata play the repeated prisoner's dilemma, Journal of Economic Theory 39, 83–96.
Selten, R., 1983, Evolutionary stability in extensive two-person games, Mathematical Social Sciences 5, 269–363.
Selten, R., 1991, Evolution, learning and economic behavior, Games and Economic Behavior 3, 3–24.
Selten, R. and R. Stoecker, 1986, End behavior in sequences of finitely repeated prisoner's dilemma supergames – A learning theory approach, Journal of Economic Behavior and Organization 7, 47–70.
Simon, H., 1982, Models of bounded rationality, 2 vols. (MIT Press, Cambridge, MA).
Smith, J., 1982, Evolution and the theory of games (Cambridge University Press, Cambridge).
Zemel, E., 1989, Small talk and cooperation: A note on bounded rationality, Journal of Economic Theory 49, 1–9.