ECONOMICS WORKING PAPERS SERIES

Economics Working Paper 51

Economic Applications of Genetic Algorithms as a Markov Process

Lisa Beth Tilis
Universitat Pompeu Fabra

June 1993

UNIVERSITAT POMPEU FABRA

Balmes, 132
Telephone (343) 484 97 00
Fax (343) 484 97 02
08008 Barcelona
e- mail econwp@upf.es

Economics Working Paper 51

Economic Applications of Genetic Algorithms as a Markov Process

Lisa Beth Tilis
Universitat Pompeu Fabra

June 1993

Keywords: Genetic Algorithm, Adaptive Systems, Learning

Journal of Economic Literature classification: D83, C13, D81

Abstract

This paper analyzes economic applications of genetic algorithms. Genetic algorithms have been studied extensively as adaptive systems capable of near optimal actions in a wide range of enviorments. It is the objective of this paper to explore how genetic algorithms may or may not be a useful tool for analyzing economic enviornments. The paper includes a brief review of other work in economics that uses genetic algorithms. The Markov representation of a simple genetic algorithm is developed, and applied to a coordination game example. This example reveals the conclusion that genetic algorithms may permit non-Nash outcomes for a significant number of periods. Further it is shown that both the short and long run behavior of a genetic algorithm is highly sensitive to the coding scheme used.

Introduction

The purpose of this paper is to analyze economic applications of genetic algorithms. Genetic algorithms have been studied extensively as adaptive systems capable of near optimal actions in a wide range of environments. It is the objective of this paper to explore how genetic algorithms may or may not be a useful tool for analyzing some specific economic environments. Section I is an introduction to genetic algorithms, their strengths and weaknesses. In addition, Section I includes a short description of current work in economics that uses genetic algorithms. Section II develops the Markov representation of a simple genetic algorithm. Section III studies an application of the algorithm to an equilibrium selection problem from game theory. This example reveals the most striking conclusion of the paper that genetic algorithms are may permit non-Nash outcomes for a significant number of periods. Section IV presents some propostions regarding the analytic properties of the algorithm. These propositions were suggested by the numerical results from Section III and serve to formalize the results from Section III. Section V is a summary and conclusions.

Section I

The genetic algorithm is a simple way to model bounded rationality of agents in economic models. It is a mathematical expression of a basic behavior rule that sounds plausible for agents to use when confronted with uncertainty and or incomplete information. The rule, stated in words, is to try strategies in proportion to their past usefulness, while continuing to accumulate information on the potential gains from as yet untried strategies. When applied to economic problems of coordination, this algorithm promises to sustain a disequilibrium state for a significant number of decision periods. This property of genetic algorithms, and other adaptive schemes suggests that using equilibrium refinements to eliminate possible solutions to game theoretic representations of economic problems may be misleading.

Adaptive schemes, such as the genetic algorithm, or best response learning, or even Bayesian updating, have ability to define the path toward a steady state. While this particular adaptive scheme has some appealing

decision theoretic properties lying at its base, as an algorithm, it offers specific rules for the decision making process that may be followed step by step. Past research on genetic algorithms suggests that it may be able to fit human subject data well. Using an adaptive process that models human behavior to construct the path toward equilibrium is much more satisfying than abstractly defining away potential equilibria on the grounds of some economists' desiderata or axioms of choice. The interesting question then is, can these models of adaptive behavior be constructed in a way so as to make them amenable to testing, if not on real world data, then on experimental data? The answer proposed here is, yes, it is possible to formalize the model so it is amenable to testing with sufficient data, but at the same time there is so much freedom in specifying the model that there may be many specifications that fit the data well.

Economists are just beginning to use genetic algorithms. There are three current papers in economics: Marimon, Sargent and McGrattan (1990) apply the genetic algorithm to the monetary model of Kyotaki and Wright. Arifovic (1989) applies the genetic algorithm to four separate smaller problems, Axelrod (1987) applies a genetic algorithm to the iterated prisoner's dilemma. Two of Arifovic's applications are overlapping generations models, one is the process of reaching competitive equilibrium and the last is a modification of Bray's (1982) model of an asset market with differentially informed agents. The current literature seems to be in general agreement over why genetic algorithms are useful for economic problems. Genetic algorithms are thought to only require two simple and intuitively appealing models of choice, agents choose to make moves according to how they have performed in the past, and agents do occasionally test new moves. These rules roughly correspond to the rules outlined in theoretical papers on learning in games. The way that the algorithm is carried out, using binary strings, assumes that agents have little knowledge of the environment that they are faced with. Depending on how the modeler sets up the algorithm, this may lead it to be very good at adapting to new environments. The testing of new moves is accomplished through a stochastic process defined by the algorithm.

There is an implicit parallelism in the structure of the algorithm that under certain specifications admits one agent considering multiple strategies at the same time. In other words it may be thought to respresent an agent holding two mutually contradictory views about the system that they are

actually in. This property may be useful in duplicating preference reservals that have appeared in the experimental literature.

A final motivation for looking further into genetic algorithms is that it is an easy way to simulate decentralized choice in an economic model. Within a game theoretic context it may help to point out how agents become focused on a particular Nash equilibrium when they exchange information in the way dictated by a genetic algorithm. However, in order to interpret any of the above simulations or applications in a meaningful way, we need to know more about the stochastic process that the rules of the algorithm define.

The Markov representation and the following example in Sections II and III will lead us to see some of the problems with genetic algorithms, and many other adaptive schemes. Perhaps the most glaring problem with using an adaptive scheme is that it is essentially arbitrary. For example in a game theoretic context one modeler may think that agents should or are thinking of mixed strategies, and thus mixed strategies should be included as possible decisions of the algorithm. Yet, an equally "correct" modeler may feel that mixed strategies are too complex for most subjects to entertain. The many different ways of parameterizing the same set of rules, is quite analogous to the many different utility functions one may entertain for describing agents' preferences in a consumer choice problem. They all are valid in some sense, but our results may differ when we use one utility function over another.

Section II

This section will focus on the properties of the stochastic process that a simple genetic algorithm defines. Previous work has been done on the expected performance of this simple genetic algorithm, but only as viewed through Holland's schemata framework. While Holland's work is important in the original formulation of the algorithm as a function optimizer, here the genetic algorithm is formalized as a Markov process. Looking at this genetic algorithm as a Markov process will yield new insights into how exactly this model of bounded rationality will play out in economic problems, thus enriching our understanding of why and how to use genetic algorithms in economics.

The genetic algorithm uses a population of n strings of bits to encode

decision rules. The interpretation that will be used in this paper is that the n strings represent n agents, each of which chooses an action as encoded by the decimal value of the string that 'is' that agent[1]. After each decision period the agents share information throught the rules described by the algorithm. These rules are the genetic operations of reproduction, crossover and mutation. After all operations have been performed, the population will be in a new state. Each period the new state will be described by the n new strings that have emerged from applying the genetic operators. The evolution of this system from one state to the next, from one set of decisions for the population to the next, is the central topic of this section.

All strings in the population have the same length, l. Thus each string may represent a decimal number in the range $[0, H]$.Where$H = 2^l - 1$. The decision that each value represents, as well as the length of the string, are parameters of the algorithm. Thus this algorithm is limited to discrete choice problems, where the maximum number of choices as $H + 1$. While each decimal value may only represent one decision, there may be a set of values which represent the same decision. The state of a string, as distinct from the state of the population, is parameterized by its decimal value. There are $H + 1$ string states.

The state of the population is parameterized by how many of the n strings are in each of the $H + 1$ string states, and thus is a vector with indexes in the range $[0, H]$. For any combination $n_1, n_2, \ldots, n_h$ such that $\sum_{i=0}^{H} n_i = n$, there is one distinct population state. The total number of possible population states, N is calculated by the number of ways to allocate n items to $H + 1$ states[2].

$$(1) \quad N = \frac{(n + H)!}{n!H!}$$

The genetic algorithm defines the three operations reproduction, crossover

[1]An alternative interpretation is that there are several populations of strings. Each population of strings then represents the decision making process of a single agent. This interpretation used in Arifovic's treatment of Bray's (1982) model of asset trading by informed and uniformed agents. The approach used in Marimon, Sargent, McGrattan is similar, althought it is one population of strings for each type of agent.

[2]This is the solution to a member of the class of problems in probability theory called occupancy problems. The classic way of formulating the problem is how many different ways are there to put n identical items into $H + 1$ distinct boxes.

and mutation to be performed in that order. In the analysis that follows it will be helpful to think of four 'stages' of the population as the algorithm is running: the original population, after reproduction, after crossover and after mutation. Once the final operator, mutation, has been applied, this is a new population.

Some notation at this point will be helpful. Let s, t, u, and v denote canonical population states. They are vectors with elements $(s_0, s_1, \ldots, s_H)$, $(t_0, t_1, \ldots, t_H)$, $(u_0, u_1, \ldots, u_H)$, $(v_0, v_1, \ldots, v_H)$. Each element denotes the number of strings within each population state that are themselves in string states $0 - H$. The vector s will be used to denote original population states, t for second stage, and u for third stage populations, and v for the final stage.

Reproduction

Reproduction has the most effect on the outcome of the algorithm as it is the operation that selects the building blocks for the final generation. Under operation of reproduction, n intermediate strings are chosen at random from the original population. The probability distribution over the original strings is that one dictated by the relative fitness (i.e. payoff, performance, utility) of the strings in the previous decision making period. Using the relative payoff of each string as its probability of being chosen for inclusion in the second stage is the means by which the most successful choices from the past are used for future decisions. The expected fitness (i.e. payoff, performance, utility) of a string in string state i within a population in state s is denoted $f[s, i]$.

$$(2) \quad F[s, i] = \frac{f[s, i]}{\sum_{k=0}^{H} f[s, k] * s_k}$$

$F[s, i]$ is the expected relative fitness (i.e. payoff, performance, utility) of a string in string state i within a population in state s.

$$(3) \quad R[s, i] = s_i * F[s, i]$$

$$(4) \quad R[s, i] = s_i * \frac{f[s, i]}{\sum_{k=0}^{H} f[s, k] * s_k}$$

$R[s,i]$ is the probability that a string in string state i will be reproduced from a population in state s.

$$(5) \quad P_{st} = \left[\prod_{i=0}^{H} R[s,i]^{t_i} \right]$$

P_{st} is the matrix of transition probabilities for going from population state s to second stage population t.

Since reproduction takes place over the original population with replacement, the order of the strings in the second stage is random, where each string from the original population has the same probability of being any one of the n second stage strings. Once the reproduction operation has been completed, n strings have been chosen for the second stage, the crossover operation is applied to the second stage.

Crossover

Crossover is an operation where two strings are mated. Then a bit position, b, is chosen with uniform probability over all l bits. The strings are then 'cut' at that position, and crossed over. In other words the front half of one string is merged with the back half of the other and vice versa. Crossover may be thought of as an information sharing device among the strings. As the strings in a population become more and more similar, the effect of the crossover operation diminishes. The extreme case is when the second stage consists on n strings all in the same string state. In this case crossover will have no effect at all. This conclusion is discussed more formally in Section IV.

Mating

The second stage has arisen in a 'random' manner, thus the pairing scheme given by mating each two adjacent strings will yield the same probability distribution over possible matings as a scheme that chooses mates over the n intermediate strings without replacement in a uniform manner. The probability that a string in string state i, and one in state j will be mated, denoted $M(i,j)$, is the same as the probability that both of these two string states end up in the second stage in either order.

$$(6) \quad M[i,j] = \left\{ \begin{array}{cc} 2 * R[s,i] * R[s,i] & i \neq j \\ R[s,i]^2 & i = j \end{array} \right\}$$

Once two strings have been paired, crossover will take place at a randomly (uniform) chosen bit, b. Let $d(i,b)$ be the decimal value of the rightmost b bits of a string in state i. By switching the rightmost b bits of string i with those of string j, the crossover operation effectively substracts $d(i,b)$ from string i, while adding $d(j,b)$. In the operation of crossover maps $\{i,j\}$ into $\{i',j'\}$.

$$(7) \quad i' = i - d(i,b) + d(j,b)$$

$$(8) \quad j' = j - d(j,b) + d(i,b)$$

Once a mating scheme is given for the n strings in the second stage, the crossover operation is completely characterized by a vector of length $n/2$ indicating which bit each pair will be crossed on. A mating scheme, A, with $a(i,j)$ pairs of i and j will have probability of occuring.

$$(9) \quad p(A) = \prod_{\substack{i=0 \,..\, H \\ j=0 \,..\, i}} M[i,j]^{a(i,j)}$$

Conditioning on a particular mating scheme, A, the set of possible states for the third stage population is limited to those that will be outcomes of $l^{n/2}$ vectors assigning $ab \in [0, l-1]$ to the $n/2$ mates. Each of these vectors will have equal probability. These vectos of crossover bits are denoted B. The operation of crossover can now be writen as $X(A,B)$.

Equation (11) gives the probability of a third stage population, u, arising from an initial population s.

$$(10) \quad I_1[A,B,u] = \left\{ \begin{array}{cc} l^{-n/2} & if\ X(A,B) \neq u \\ 0 & otherwise \end{array} \right\}$$

$$(11) \quad P_{su} = \begin{array}{l} \sum_A [p(A) * \sum_B I_1(A,B,u)] \\ A \in \{mating\ schemes\} \\ B \in \{crossover\ vectors\} \end{array}$$

Even for small population sizes, and short string lengths, calculating this matrix involves checking many mating schemes and crossover vectors[3].Section III includes an example of a two string, two bit system. In this small example alone there are ten population states, one mating scheme and two possible crossover vectors[4].

Mutation

The mutation operation will, with some small probability, ϵ, switch any bit of any string in the entire population. There is a Markov transition probability matrix associated with this operation. The entries in this matrix represent the probability that mutation will bring a third stage in state u, into a final stage v. The first thing to note in deriving these probabilites is that any string in the third stage population can be mutated to be any one of the strings of the final stage population. The probability that a given string i from population u will be mutated enough to become string j from population v is

$$(12)\quad m(i,j) = \varepsilon^{\sum_{b=0}^{l-1} I_2(b)} * (1-\varepsilon)^{l-\sum_{b=0}^{l-1} I_2(b)}$$

Note $m(i,j) > 0$ for all i,j.

Where $I_2(b)$ is an indicator function that takes on the value one when the two corresponding bits of strings i and j have different values, and zero when they are the same value[5]. Notice that $\sum I_2(b)$ is now a distance measure showing how far in bits it is between strings i and j.

$$(13)\quad I_2(b) = \left\{ \begin{array}{ll} 1 & d(i,b) - d(i,b-1) = d(j,b) - d(j,b-1) \\ 0 & otherwise \end{array} \right\}$$

---

[3]There are $l^{n/2}$ crossover vectors, and the number of mating schemes is $1/2\left((H+1)^2 + (H+1)\right)$. The number of crossover vectors is the number of ways to assing up to l items in each of $n/2$ boxes. The number of mating schemes was figured as the number of elements in the lower triangle (including the diagonal) of a matrix with $H+1$ rows and $H+1$ columns. Basically this is just the area of a triangle with base $= H+1$ and height $= H+1$, plus a small half triangle for each diagonal element.

[4]In this system the crossover vector is actually a scalar indicating where the single pair of strings in the population is to be crossed.

[5]This distribution differs from the binomial because the order of the 'successes' is important. Thus, there is no binomial coefficient.

$$(14)\quad D(i,j) = \sum_{b=0}^{l-1} I_2(b)$$

If there are k_s distinct states represented in s, and k_v distinct states represented in v, then Equation (15) gives the number of possible matchings. Each of these matchings imply a probability of the mutation operation making u mutate into v. This probability is given in Equation 16[6].

$$(15)\quad K = \frac{(k_s + k_v - 1)!}{k_s!(k_v - 1)!} k_s < k_v$$

$$(16)\quad h(u,v) = \prod_{i=i}^{n} m(u^i, v^i)$$

$$(17)\quad P_{uv} = \left[\sum_k h^k(u,v)\right]$$

In (16) i indexes the string number for the particular matching. In (17) k sums over all $n!$ possible matchings, thus P_{uv} is the matrix for mutation alone. Now the entire genetic algorithm may be characterized by the product of P_{su} and P_{uv}.

$$(18)\quad P_{sv} = P_{su} * P_{uv}$$

Section III

This section presents a simple example of a genetic algorithm. The example is a two person matrix game that was used in Cooper, DeJong, Forsythe, Ross (1990). The particular game that this section addresses is Game 3 from their paper. It is an example of a coordination game with two Pareto rankable pure strategy Nash equilibria, one mixed strategy Nash equilibrium, as well

[6]When n is large it may be computationally more efficient to consider how many of these matchings are redundant due to more than one string being in a particular string state in either population.

as a cooperative outcome. The payoff matrix is presented in Figure 1. The entry in each cell is the row player's payoff. It is understood as follows. If the row player plays $R1$ and the column player plays $C1$ then the row player wins 350 points. The game is symmetric, so that the column player's payoffs are the same for a given action by the row player, as the row player's payoff are given the same action by the column player.

The fact that this example contains multiple equilibria as well as a cooperative outcome makes it an interesting testing ground for the equilibrium selection properties of genetic algorithms.

A two string ($n = 2$), two bit ($l = 2$) system is used to calculate the transition matrices for a genetic algorithm. This is a much smaller system than those presented in previous work on genetic algorithms, however the main method and results are much clearer in this environment than in larger systems. Each of the two strings represents one of the players of the game. With two bits per string the algorithm provides four possible decisions for each of the players.

Figure 2 shows six possible specifications for this system[7]. The mixed strategy was chosen to correspond to the mixed strategy Nash equilibrium. Each specification uses the same set of possible actions. Figure 3 shows the mapping from actions to population states. This mapping is used for all codings. Figure 4 shows the reproduction and crossover matrices for all 6 specifications. Figure 5 shows the matrix for the entire process for all six specifications. All the probabilities given are based on the payoffs from Figure 1 and the codings in Figure 2.

The six specifications do in fact have different Markov representations. Figure 6 shows that the long run distributions over states for these six specifications are also different across codings. Codings A and C have the same reproduction and crossover matrices, but mutation makes the processes differ in the short run. Their long run distributions however are the same. It is encouraging to note that although the long run distributions are sensitive to coding, the difference between them seems to be minimal.

Much of the previous work on genetic algorithms, both within and outside of economics, has allowed the probability of mutation to diminish as time goes

[7]The set of six specifications is the set of permutations of the last three strings in the ordering of the mapping from string states to actions.

on. The reason for this becomes clear when comparing Figures 4 and 5. In Figure 4, we see that there are four absorbing states of the system. States 1,4,7, and 9 will all be long run stationary points of the algorithm once they are entered into. However, if mutation is introduced, there are no longer any absorbing states of the system. In other words the algorithm will never settle down to one state, but rather it will settle to a long run distribution over states. The long run distributions for this system are given in figure 6. By diminishing the probability of mutation, the algorithm will converge to one of the absorbing states with probability one. Intuitively this is because eventually it will become the system described in Figure 4.

Looking at Figure 4, we can see that there is positive probability that the system, without mutation, could settle down to the cooperative outcome, state 9. Here, the cooperative outcome, as well as all three Nash equilibria are states where all the strings are the same. When all strings are the same, crossover and reproduction have no effect on the system, thus an absorbing state arises. It is the symmetry of the system that brings the it to a Nash equilibrium, not the payoffs of the game.

Section IV

This section will present some analytic properties of the simple algorithm discussed thus far. Three propositions that give some more insight into how the crossover and mutation operations work are stated and proven. A brief example of why coding matters even in systems with longer strings is given. The section also includes some propositions leading to a proof of the ergodicity of the Markov process defined by the algorithm. This leads to some conclusions regarding the stability of both Nash and non-Nash outcomes of the algorithm. The section is concluded with suggestions for future research.

The effect of crossover in different systems is expressed in the following propositions. Propositions 1 and 2 say that the effect of crossover diminishes as the two strings to be crossed are more similar. Proposition 1 is stated in terms of the decimal representation of the strings. Proposition 2 is stated in terms of the 'bit distance' as defined by $I_2(b)$. The conclusion from Proposition 2 says that the overall change in bit distance possible from a given crossover operation gets smaller as two strings become closer in bit distance.

Propositions 3 and 4 say that as the length of the strings in a system gets smaller, the effect of crossover alone diminishes.

Proposition 1 *As* $|d(i,b) - d(j,b)| \to 0 \qquad |i - i'| \to 0 \qquad$ *monotonically*

Proof 1: rearrange Equation (7)

$$i - i' = d(i,b) - d(j,b)$$

Proposition 2 As $D(i,j) \to 0$

$$\max \qquad \{D(i,i') + D(j,j'), D(i,j') + D(j,i')\} \to 0$$

Proof 2:

$$\max \qquad D(k,k') = D(i,j) - 1 \quad k = i,j \;\; k' = i',j'$$

As $D(i,j) \to 0 \quad \max \quad D(k,k') \to 0$

Proposition 3 As $l \to 0 |i - i'| \to 0$

Proof 3:

$$\max_b \qquad d(i,b) = d(i,l)$$

$$\min_b \qquad d(j,b) = d(j,0) = 0$$

as $l \to 0 \qquad \max_b \qquad d(i,b) \to 0$ monotonically

as $l \to 0 \qquad \max_b \qquad |d(i,b) - d(j,b)| \to 0$ monotonically

therefore by Proposition 1 $\qquad |i - i'| \to 0$

Proposition 4 *As* $l \to 0$

$$\max \quad \{D(i,i') + D(j,j'), D(i,j') + D(j,i')\} \to 0$$

Proof 4: directly from Proposition 2

and as $l \to 0 \qquad D(i,j) \to 0$

Taken together these propositions suggests that the potential difference from recoding would only become more stark in larger systems, precisely because crossover would be a stronger operation. The four bit recoding example given in figure 7 supports this conjecture. In this example each bit may be taken to indicate an action at a particular information set. The first line gives a mated pair and the four possible outcomes of crossover. The second line recodes the first by switching bits 1 and 3 and 2 and 4. Again all four possible outcomes are given. Finally the third line recodes the second back to the same order as the first. It is clear that the possible outcomes of the crossover operation are different under different coding schemes. Mutation is also affected by different coding schemes. When a coding scheme changes the bit distance between two actions changes, then obviosly the probability that one will mutate into the other is changed.

Propositions 5 and 6 serve to set up notation and concepts for the last three propositions. Proposition 7 characterizes the Markov process when the mutation probability remains constant. One important conclusion to be drawn from Proposition 7 is that as long as the mutation probability is positive, all states will be visited infinitely often. Notice however that all states have this property, including, but not limited to the Nash equilibria.

Proposition 7 also indicates the existence of a unique long run stationary distribution over states. This result follows directly from the ergodicity of a finite state Markov chain. This fact leads to the possibility that the long run distribution may the same for some or all permutations of the coding scheme. A more general characterization of when this could be true is a topic for further research.

Finally Proposition 8 is a preliminary way of characterizing the behavior of the system if the mutation probability shrinks over time. Recall that one of the conclusions from Section III was that if there were no mutation at all, then

all symmetric states would be long run stationary points of the algorithm. Proposition 7 then tells us that even if ϵ is allowed to be positive, it can be chosen so that the probability of leaving a symmetric state, Nash or non-Nash, is close to zero. This conclusion naturally leads to further questions regarding the convergence properties of the algorithm when ϵ shrinks. This would be a non-stationary Markov process, and its precise long run behavior is another topic for future research.

Proposition 5 *If $\epsilon > 0$, then P_{uv} is a strictly positive matrix.*

Proof 5:

$m(i,j) > 0$ for all i,j	from Eq. 12
$h^k(u,v) > 0$ for all u,v,k	from Eq. 15
denoting the element of P_{uv} as $q(u,v)$	
$q(u,v) > 0$	from Eq. 16

Proposition 6 *If P_{uv} is a strictly positive matrix then P_{sv} is a strictly positive matrix.*

Proof 6:

$P_{sv} = P_{su} * P_{uv}$ from Eq. 17

denoting the elements of these matrices as $r(s,v), p(s,u)$ and $q(u,v)$ respectively gives

$$r(s,v) = \sum_{u=0}^{N-1} p(s,u) \text{ x } q(u,v)$$

$$p(s,u) \geq 0$$

$$and \sum_{u=0}^{N-1} p(s,u) = 1$$

$$\therefore \exists\ u\ such\ that\ p(s,u) > 0$$

$q(u,v) > 0 \forall\ u,v$ *by Proposition* 5

$\therefore r(s,v) > 0$

Proposition 7 *The Markov process for the entire genetic algorithm is irreducible, aperiodic and all states are positive recurrent.*

Proof: Irreducibility comes from the fact that the matrix is stricly positive therefore all states communicate.

The fact that the matrix is strictly positive also makes the process aperiodic as this implies all diagonal elements are positive, thus each state can return to itself in any number of periods.

All states must be positive recurrent, since they are all in one recurrence class and this is a finite state Markov process.

$$For\ t = 1,2,3\ ....\ \exists \quad \varepsilon > 0 \quad | \quad P_{sv}^{t} < \delta, \delta > 0$$

Proposition 8 $\forall v, \Lambda$

$\forall\ s\ such\ that\ s_i = n\ for\ some\ i$

Proof: Proof follows directly from the fact that powers of ϵ enter into the system additively when P_{su} is multiplied by P_{uv}. Equations 12 and 15 show that only powers of ϵ are in P_{uv}.

Section V

This paper has shown that the genetic algorithm can be formalized as a Markov process. The Markov representation of a genetic algorithm with the operators of reproduction, crossover and mutation was presented for a general system. When formalized as a Markov process genetic algorithms have the potential to be testable and or calibrated to experimental data. However, the study of an application of the algorithm presented in Section III illustrates

the sensitivity of the algorithm to coding schemes. It was shown that both short run and long run behavior of the algorithm may differ under different coding schemes. It was shown that absorbing states of the system are purely an artifact of the coding scheme, and that only outcomes that are symmetric with respect to string states will be absorbing states of the system. Section IV presented several propositions to formalize the results from Section III. So, while the algorithm is an intuitively appealing adaptive scheme, care must be taken to note the potential bias from the coding scheme.

Future research on genetic algorithms could include a closer study of the model when the mutation probability diminishes over time. Also other modifications of the algorithm, such as a bucket brigade payoff assignment, could be studied. Other future research could include a study of how rational agents might affect the outcome of the algorithm. Topics of this nature could be related to current work on replicator dynamics and evolutionary stability. This paper also suggests that more detailed research into the existing models of learning could prove enlightening. For example these studies use learning algorithms like the genetic algorithm, and they conclude that Nash equilibrium will be reached infinitely often, but they don't clearly state if other outcomes will also be reached infinitely often.

References

ARIFOVIC, J. (1989). *Learning by Genetic Algorithms in Economic Environments.* PhD thesis, University of Chicago.

AXELROD, R. (1987). *Genetic Algorithms and Simulated Annealing*, capítulo The Evolution of Strategies in the Iterated Prisoner's Dilemma. Morgan Kaufman, Los Altos, California. Lawrence Davis eds.

BOOKER, L. B., D. E. GOLBERG, AND J. H. HOLLAND (1989), "Classifier Systems and Genetic Algorithms," *Artificial Intelligence*, 40:235–282.

BRAY, M. (1982), "Learning, Estimation, and the Stability of Rational Expectations," *Journal of Economics Theory*, 26(2):318–339.

BRAY, M. M. AND N. E. SAVIN (1986), "Rational Expectations Equilibria, Learning and Model Specification," *Econometrica*, 54(5):1129–1160.

COOPER, R., D. DEJONG, R. FORSYTHE, AND T. ROSS (1990), "Selection Criteria in Coordination Games: Some Experimental Results," *American Economic Review*, pages 234–248.

DEJONG, K. (1980), "Adaptive System Design: A Genetic Approach," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC–10(9).

FUDENBERG, D. AND D. KREPS (1989). "A Theory of Learning, Experimentation and Equilibrium Selection in Games, " Unpublished manuscript.

GOLDBERG, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison–Wesley, Menlo Park, California.

KARLIN, S. AND HOWARD TAYLOR (1981). *A Second Course in Stochastic Processes.* Academic Press, New York.

LUCAS, R. (1986), "Adaptive Behavior and Economic Theory," *Journal of Business*, 59:5401–5426.

MARIMON, R., E. MCGRATTAN, AND T. SARGENT (1990), "Money as a Medium of Exchange in an Economy with Artificially Intelligent Agents," *Journal of Economic Dynamics and Control*, 14:329–373.

MILGROM, P. AND J. ROBERTS (1990). "Rationalizability, Learning and Equilibrium in Games with Strategic Complementarities, " Technical report, Stanford University. Working Paper.

ROSS, S. (1983). *Stochastic Processes.* Wiley, New York.

SELTEN, R. (1989). "Evolution, Learning and Economic Behavior, " Technical report, Kellogg Graduate School of Management. Northwestern University. Nancy L. Schwartz Memorial Lecture, J. L.

Table 1: Payoff Matrix

	C1	C2	C3
R1	350	350	1000
R2	250	550	0
R3	0	0	600

Table 2: Six Codings

String	A	B	C	D	E	F
00	mix	mix	mix	mix	mix	mix
01	1	1	2	2	3	3
10	2	3	1	3	1	2
11	3	2	3	1	2	1

Table 3: Actions to Population States

Action for String i	Action for String j	Population State
mix	mix	0
mix	1	1
mix	2	2
mix	3	3
1	1	4
1	2	5
1	3	6
2	2	7
2	3	8
3	3	9

Table 4: Continued

Coding E									
100.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
22.6	49.9	0.00	0.00	27.6	0.00	0.00	0.00	0.00	0.00
29.5	0.00	24.8	0.00	0.00	0.00	24.8	20.8	0.00	0.00
100.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	100.	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	34.0	48.6	0.00	17.4	0.00	0.00
0.00	0.00	0.00	0.00	100.	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	25.0	50.0	25.0
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.
Coding F									
100.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
22.6	24.9	0.00	0.00	27.6	0.00	0.00	0.00	24.9	0.00
29.5	0.00	49.6	0.00	0.00	0.00	0.00	20.8	0.00	0.00
100.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	100.	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	34.0	48.6	0.00	17.4	0.00	0.00
0.00	0.00	0.00	0.00	100.	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.	0.00	0.00
0.00	25.0	0.00	0.00	0.00	0.00	0.00	25.0	25.0	25.0
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.

Table 5: Markov Matrix for Reproduction, Crossover and Mutation

Coding A									
41.0	20.5	20.5	5.10	2.60	5.10	1.30	2.60	1.30	0.200
15.1	32.0	7.54	8.00	17.0	8.00	8.50	0.942	2.00	1.06
17.7	8.85	31.9	7.97	1.11	7.97	1.99	14.4	7.18	0.897
41.0	20.5	20.5	5.10	2.60	5.10	1.30	2.60	1.30	0.200
2.60	20.5	1.30	5.10	41.0	5.10	20.5	0.200	1.30	2.60
2.60	12.5	9.31	13.8	15.2	13.8	12.5	8.47	9.31	2.60
2.60	20.5	1.30	5.10	41.0	5.10	20.5	0.200	1.30	2.60
2.60	1.30	20.5	5.10	0.200	5.10	1.30	41.0	20.5	2.60
1.00	2.00	8.00	8.00	1.00	8.00	8.00	16.0	32.0	16.0
0.200	1.30	1.30	5.10	2.60	5.10	20.5	2.60	20.5	41.0
Coding B									
41.0	20.5	5.10	20.5	2.60	1.30	5.10	0.200	1.30	2.60
15.1	32.0	8.00	7.54	17.0	8.50	8.00	1.06	2.00	0.942
13.4	11.7	14.0	11.7	2.60	10.1	14.0	9.88	10.1	2.60
41.0	20.5	5.10	20.5	2.60	1.30	5.10	0.200	1.30	2.60
2.60	20.5	5.10	1.30	41.0	20.5	5.10	2.60	1.30	0.200
1.21	9.67	7.92	1.98	19.3	31.7	7.92	13.0	6.49	0.812
2.60	20.5	5.10	1.30	41.0	20.5	5.10	2.60	1.30	0.200
0.200	1.30	5.10	1.30	2.60	20.5	5.10	41.0	20.5	2.60
1.00	2.00	8.00	8.00	1.00	8.00	8.00	16.0	32.0	16.0
2.60	1.30	5.10	20.5	0.200	1.30	5.10	2.60	20.5	41.0

Table 5: Continued

Coding C									
41.0	20.5	20.5	5.10	2.60	5.10	1.30	2.60	1.30	0.200
15.1	32.0	7.54	8.00	17.0	8.00	8.50	0.942	2.00	1.06
17.7	8.85	31.9	7.97	1.11	7.97	1.99	14.4	7.18	0.897
41.0	20.5	20.5	5.10	2.60	5.10	1.30	2.60	1.30	0.200
2.60	20.5	1.30	5.10	41.0	5.10	20.5	0.200	1.30	2.60
2.60	12.5	9.31	13.8	15.2	13.8	12.5	8.47	9.31	2.60
2.60	20.5	1.30	5.10	41.0	5.10	20.5	0.200	1.30	2.60
2.60	1.30	20.5	5.10	0.200	5.10	1.30	41.0	20.5	2.60
1.00	2.00	8.00	8.00	1.00	8.00	8.00	16.0	32.0	16.0
0.200	1.30	1.30	5.10	2.60	5.10	20.5	2.60	20.5	41.0
Coding D									
41.0	5.10	20.5	20.5	0.200	1.30	1.30	2.60	5.10	2.60
10.6	14.1	10.4	10.4	12.7	11.4	11.4	2.60	14.1	2.60
17.7	7.97	31.9	8.85	0.897	7.18	1.99	14.4	7.97	1.11
41.0	5.10	20.5	20.5	0.200	1.30	1.30	2.60	5.10	2.60
0.200	5.10	1.30	1.30	41.0	20.5	20.5	2.60	5.10	2.60
0.812	7.92	6.49	1.98	19.3	31.7	9.67	13.0	7.92	1.21
0.200	5.10	1.30	1.30	41.0	20.5	20.5	2.60	5.10	2.60
2.60	5.10	20.5	1.30	2.60	20.5	1.30	41.0	5.10	0.200
2.60	14.1	10.9	10.9	2.60	10.9	10.9	11.6	14.1	11.6
2.60	5.10	1.30	20.5	2.60	1.30	20.5	0.200	5.10	41.0
Coding E									
41.0	20.5	5.10	20.5	2.60	1.30	5.10	0.200	1.30	2.60
15.1	32.0	8.00	7.54	17.0	8.50	8.00	1.06	2.00	0.942
13.4	11.7	14.0	11.7	2.60	10.1	14.0	9.88	10.1	2.60
41.0	20.5	5.10	20.5	2.60	1.30	5.10	0.200	1.30	2.60
2.60	20.5	5.10	1.30	41.0	20.5	5.10	2.60	1.30	0.200
1.21	9.67	7.92	1.98	19.3	31.7	7.92	13.0	6.49	0.812
2.60	20.5	5.10	1.30	41.0	20.5	5.10	2.60	1.30	0.200
0.200	1.30	5.10	1.30	2.60	20.5	5.10	41.0	20.5	2.60
1.00	2.00	8.00	8.00	1.00	8.00	8.00	16.0	32.0	16.0
2.60	1.30	5.10	20.5	0.200	1.30	5.10	2.60	20.5	41.0
Coding F									
41.0	5.10	20.5	20.5	0.200	1.30	1.30	2.60	5.10	2.60
10.6	14.1	10.4	10.4	12.7	11.4	11.4	2.60	14.1	2.60
17.7	7.97	31.9	8.85	0.897	7.18	1.99	14.4	7.97	1.11
41.0	5.10	20.5	20.5	0.200	1.30	1.30	2.60	5.10	2.60
0.200	5.10	1.30	1.30	41.0	20.5	20.5	2.60	5.10	2.60
0.812	7.92	6.49	1.98	19.3	31.7	9.67	13.0	7.92	1.21
0.200	5.10	1.30	1.30	41.0	20.5	20.5	2.60	5.10	2.60
2.60	5.10	20.5	1.30	2.60	20.5	1.30	41.0	5.10	0.200
2.60	14.1	10.9	10.9	2.60	10.9	10.9	11.6	14.1	11.6
2.60	5.10	1.30	20.5	2.60	1.30	20.5	0.200	5.10	41.0

Table 6: Long Run Distributions

	Coding					
State	A	B	C	D	E	F
0	14.6	13.4	14.6	12.5	13.4	12.5
1	16.8	16.8	16.8	7.16	16.8	7.16
2	12.8	6.77	12.8	13.4	6.77	13.4
3	6.74	8.61	6.74	8.59	8.61	8.59
4	14.4	15.6	14.4	13.7	15.6	13.7
5	6.74	13.4	6.74	14.1	13.4	14.1
6	9.14	6.77	9.14	9.17	6.77	9.17
7	7.46	7.52	7.46	10.0	7.52	10.0
8	7.12	7.09	7.12	7.16	7.09	7.16
9	4.15	4.04	4.15	4.22	4.04	4.22

Table 7: Four Bit Recoding Example

				Cross bit					
		String	String	b=1		b=2		b=3	
		i	j	i'	j'	i'	j'	i'	j'
original	0101	1110	0101	1100	0100	1111	0110	1101	0110
recode	0101	1011	0101	1011	0101	1011	0111	1001	0011
decode	0101	1110	0101	1110	0101	1110	1101	0110	1100

Appendix

```
Program to do calculations for Section III
Note: Function Action changed for each coding.
Program GA2by2;
Uses Crt, Dos;

Const
   n = 2; {number of strings}
   l = 2; {number of bits}
   H = 3; {2^l-1, highest decimal number representable}
   BigN = 10; { (n+H)!/n!H! number of population states}
   Eps = 0.2; {mutation probability}
Type
  PopStateType = array[0..H] of Byte;

Var
 Popstates : array[0..BigN-1] of PopState Type;
 Tostate : PopStateType;

 Payoffs : array [0..BigN-1, 0..H] of real;

 Mates : array[0..BigN-1, 0..l-1] of record
    Iprime, JPrime : Byte;
    end;

 TotalFit : Real;

 RelFit : Array [0..BigN-1, 0..H] of real, {relative fitness}
 Reprob : Array [0..BigN-1, 0..H] of Real; {prob. of reproduction}
 Mateprob: Array [0..BigN-1, 0..H, 0..H] of real; {prob. of mating}

 SumReprob, SumRelfit : REal; {used for check if Reprob}

 Popst, Act, StrstI, StrstJ : Integer ;
 MateNum, Crossbit, ToStates : Integer;
 PrimeNum, Strstipri, Strstjpri : Integer;

 IONE : Array [0..BigN-1, 0..l-1, 0..BigN-1] of real;
 {indicator for if mating scheme A and crossover vector B will yield
 population u}

 SumOverB: Array [0..Bign-1] of Real; {used to sum IONE over crossover
 vectors}

 BigPSU : Array [0..BigN-1, 0..BigN-1] of real;
 {The reproduction + crossover Matrix}

 SumOverPopSt : Real; {used to check that things sum to one}

 Mute : Array [0..H, 0..H] of real;
 {probability of string i mutating into string j}

 BigPUV : Array[0..BigN-1, 0..BigN-1] of real;
 {The mutation Matrix}
```

```
{OUTPUT FILE}
 DataFile :Text;
 DataFileName :String[40];
{TIME/DATE}
 Hour, Min, Sec, Sec100 : word;
 Month, Day, Year, DayofWeek : word;

{*****************  YES  ********************************************}

function Yes( Prompt : string ) : boolean;

var
 Ch : char;

begin

 write( Prompt, '(Y/N)');
 repeat
  Ch := readkey;
 until Ch in ['Y', 'y', 'N', 'n' ];
 write( UpCase( Ch ) );
 Yes := ( Ch in ['Y', 'y'] );

 Writeln;

end; { Yes }

{******************  OPENDataFILE  ************************************}
{Opens, Signs and Dates the DataFile}
Procedure OpenDataFile;
  Var
   OK : boolean;
  Begin
   repeat
    Writeln;
    Write ('Enter name of Data File ');
    Readln(DataFileName);
    Writeln;
    Assign(DataFile, DataFileName);

    {$I-} ReSet(DataFile) {$I+};
    OK := (IOresult = 0);

    If OK then begin
       Close(Datafile);
       Writeln('The File',DataFileName,'already exists.');

       If Not Yes('Do you want to write over this file?') then
          OK := False;
    End{if}
    Else
       OK := true;
    until OK;
```

```
    ReWrite(Datafile);
    Writeln(DataFile,'This is file',DatafileName);

    Writeln(Datafile,'The Output contained herein was generated by
    GA2by2');
    Writeln(Datafile,'As written By Lisa Tilis');

    GetTime(Hour,Min,Sec,Sec100);
    Writeln(Datafile,'Time:',Hour:2,':',Min:2);

    GetDate(Year,Month,DAy,DayofWeek);
    Writeln(Datafile,'Date:',Month:2,'/',Day:2,'/',Year:4);

    Writeln(DataFile);
    Writeln(DataFile,'-------------------------------------------------------');

  End; {proc OpenDataFile}
{******************  CLOSEFILES  ************************************}
{Closes and dates all open files}

Procedure CloseFiles;
  Begin
    GetTime(Hour,Min,Sec,Sec100);
    GetDate(Year,Month,DAy,DayofWeek);


    Writeln(Datafile,'Time:',Hour:2,':',Min:2);
    Writeln(Datafile,'Date:',Month:2,'/',Day:2,'/',Year:4);


    Close(DataFile);

  End;
Function Action (StrSt:Byte):Byte;
{Takes in the decimal value of a string State and returns the action #}

 Begin
   Case Strst of
    0:Action:=0;
    1:Action:=3;
    2:Action:=2;
    3:Action:=1;
  end; {case}
 End; {function Action}

Function StateEqual( State1, State2 : PopStateType):Boolean;
 Var
  i:integer;

 Begin
   StateEqual:= True;
   For i:=0 to H do
    If State1[i] <> State2[i] then
       StateEqual :=False;
```

```
 End; {Function StateEqual}

Function Stateof( Strst1, Strst2 : integer): integer;
 Var
  State : Popstate Type;
  i, temp: integer;

 Begin

  For i:=0 to H do
   State[i] :=0;

  State[ Action(Strst1) ] :=State[ Action(Strst1) ] + 1;
  State[ Action(Strst2) ] :=State[ Action(Strst2) ] + 1;

  Temp := BigH;

  For i := 0 to BigH -1 do
   If StateEqual(State,Popstates[i]) then
      Temp :=i;

  If Temp = BigH then
     Writeln ('error in stateof')
  Else Stateof := Temp;

End; {function stateof}


Procedure InitVariables;
 Begin
   For Popst := 0 to BigH-1 do
     For Act := 0 to H do begin
       Popstates[Popst, Act] := 0;
       Payoffs[Popst,Act] := 0;
     end; {for}

   Popstates[0,0] := 2; Payoffs[0,0] := 350;
   Popstates[1,0] := 1; Payoffs[1,0] := 950/3;
   Popstates[2,0] := 1; Payoffs[2,0] := 1250/3;
   Popstates[3,0] := 1; Payoffs[3,0] := 2000/3;

   Popstates[1,1] := 1; Payoffs[1,1] := 350;
   Popstates[4,1] := 2; Payoffs[4,1] := 350;
   Popstates[5,1] := 1; Payoffs[5,1] := 350;
   Popstates[6,1] := 1; Payoffs[6,1] := 1000;

   Popstates[2,2] := 1; Payoffs[2,2] := 350;
   Popstates[5,2] := 1; Payoffs[5,2] := 250;
   Popstates[7,2] := 2; Payoffs[7,2] := 550;
   Popstates[8,2] := 1; Payoffs[8,2] := 0;

   Popstates[3,3] := 1; Payoffs[3,3] := 0;
   Popstates[6,3] := 1; Payoffs[6,3] := 0;
```

```
Popstates[8,3] := 1; Payoffs[8,3] := 0;
Popstates[9,3] := 2; Payoffs[9,3] := 600;

Mates[0,0].Iprime := 0;
Mates[1,0].Iprime := 0;
Mates[2,0].Iprime := 0;
Mates[3,0].Iprime := 0;
Mates[4,0].Iprime := 1;
Mates[5,0].Iprime := 1;
Mates[6,0].Iprime := 1;
Mates[7,0].Iprime := 2;
Mates[8,0].Iprime := 2;
Mates[9,0].Iprime := 3;

Mates[0,0].Jprime := 0;
Mates[1,0].Jprime := 1;
Mates[2,0].Jprime := 2;
Mates[3,0].Jprime := 3;
Mates[4,0].Jprime := 1;
Mates[5,0].Jprime := 2;
Mates[6,0].Jprime := 3;
Mates[7,0].Jprime := 2;
Mates[8,0].Jprime := 3;
Mates[9,0].Jprime := 3;

Mates[0,1].Iprime := 0;
Mates[1,1].Iprime := 1;
Mates[2,1].Iprime := 0;
Mates[3,1].Iprime := 1;
Mates[4,1].Iprime := 1;
Mates[5,1].Iprime := 0;
Mates[6,1].Iprime := 1;
Mates[7,1].Iprime := 2;
Mates[8,1].Iprime := 3;
Mates[9,1].Iprime := 3;

Mates[0,1].Jprime := 0;
Mates[1,1].Jprime := 0;
Mates[2,1].Jprime := 2;
Mates[3,1].Jprime := 2;
Mates[4,1].Jprime := 1;
Mates[5,1].Jprime := 3;
Mates[6,1].Jprime := 3;
Mates[7,1].Jprime := 2;
Mates[8,1].Jprime := 2;
Mates[9,1].Jprime := 3;

Mute[0,0] := (1-Eps)*(1-Eps);
Mute[0,1] := (Eps)*(1-Eps);
Mute[0,2] := (Eps)*(1-Eps);
Mute[0,3] := Eps*Eps;

Mute[1,1] := (1-Eps)*(1-Eps);
```

```
Mute[1,2] := (Eps)*(Eps);
Mute[1,3] := (Eps)*(1-Eps);

Mute[2,2] := (1-Eps)*(1-Eps);
Mute[2,3] := (Eps)*(1-Eps);

Mute[3,3] := (1-Eps)*(1-Eps);

For Strsti := 0 to H do
  For Strstj := strsti to H do
    Mute[strstj,strsti] := Mute [strsti,strstj];

Writeln(DATAFILE, 'Popstates');
For PopSt := 0 t0 biGn -1 D0 begin
  Write (DATAFILE,Popst:4);
  For Act := 0 to H do
    write(DATAFILE, PopSates[PopSt, Act]:4);
  Writeln(DATAFILE);
End;
Writeln(DATAFILE);

Writeln(DATAFILE,'Payoffs');
For PopSt := 0 t0 biGn -1 D0 begin
  Write (DATAFILE,Popst:4);
  For Act := 0 to H do
    write(DATAFILE,Payoffs[PopSt,Act]:5:0);
  Writeln(DATAFILE);
End;
Writeln(DATAFILE);

Writeln(DATAFILE,'Mates     b=0      b=1  ');
Writeln(DATAFILE,'State IPRIME JPRIME IPRIME JPRIME');
For PopSt := 0 t0 biGn -1 D0 begin
  Write(DATAFILE,Popst:6);
  For Crossbit := 0 to l-1 do
    write(DATAFILE,Mates[PopSt,Crossbit].IPRIME:6,Mates[PopSt,Crossbit].JPRIME:6);
  Writeln(DATAFILE);
End;

Writeln(DataFile);
Writeln(DataFile,'Mutation Probabilities string to string');

For Strsti := 0 to H do
  Write(Datafile, Strsti:8);
Writeln(DATAFILE);
For Strsti := 0 to H do begin
  Write(Datafile, Strsti:4);
  For Strstj := 0 to H do
    Write(DATAFILE,Mute[Strsti,Strstj]*100:7:1);

  Writeln(DATAFILE);
End; {for}
```

```
    End;{Procedure initvariables}

   Begin
    OpenDataFile;

    Writeln(DataFile,'Strst',' Action');
    For Strsti := 0 to H do
      Writeln(DataFile,Strsti:5,Action(Strsti):5);
    InitVariables;

    For Popst := 0 to BigM-1 do Begin
      TotalFit := 0;

      For Strsti := 0 to H do
        TotalFit := (Payoffs[Popst,Action(StrSti)]*
 Popstates[Popst,Action(strsti)]) + TotalFit;

     Write(DATAFILE,'Total Fitness is..',TotalFit:5:0,' for state',Popst:4);

     SumRelFit := 0;
     SumReprob := 0;
     For Strsti := 0 to H do begin
      If TotalFit > 0 then
RelFit [PopSt, Strsti] := Payoffs[Popst,Action(Strsti)]/TotalFit
      Else
RelFit[PopSt,StrSti] := (1/n);

      Reprob[PopSt,Strsti] := (Popstates[Popst,Action(strsti)])*
        RelFit[PopSt,SrtSti];

      SumRelFit := SumRelfit +
       (RElFit [Popst,StrSti] * Popstates[Popst,Action(strsti)]);

      SumReprob := SumReprob + Reprob[PopSt,Strsti];

    End; {for StrSti}
    Writeln (DATAFILE,' SumReprob',SumReprob:7:2,'
    SumRelfit',SumRelfit:7:2);

    For StrSti := 0 to H do begin
      For StrStj := 0 to (StrSti-1) do begin
        MateProb[PopSt,Strsti,strstj] := 2*Reprob[popst,strsti]*
     Reprob[popst,strstj];
        MateProb[PopSt,Strstj,Strsti] := MateProb[Popst,Strsti,Strstj];
      End; {for}

      MateProb[Popst,Strsti,Strsti] :=
      Reprob[popst,strsti]*Reprob[popst,strsti];
    End; {for}

 End; {for popst}
{Set up IONE}
  Writeln(DataFile);
  Writeln(DataFile,'IONE');
```

```
 Writeln(DataFile,'Mate','Cross','To');

 For MateNum := 0 to BigN-1 do
  For Crossbit := 0 to l-1 do begin
    For Act := 0 to H do
      ToState[Act] := 0;
{could use stateof here}
      ToState[ Action(Mates[ MateNum, Crossbit].Iprime)]:=
      ToState[ Action(Mates[ MateNum, Crossbit].Iprime)] + 1;

      ToState[ Action(Mates[ MateNum, Crossbit].Jprime)] :=
      ToState[ Action(Mates[ MateNum, Crossbit].Jprime)] + 1;

      For ToStates := 0 to BigN-1 do
        If StateEqual(ToState,PopStates[ Tostates]) then begin
   IONE [ Matenum,crossbit,tostates] := 0.5;
   Writeln (DATAFILE,MateNum:3,Crossbit:5;ToState:5);
end{if}
Else
   IONE [Matenum,crossbit,tostates] := 0;

   End; {for}
{Done setting up IONE}

{Note here the probability of a mating scheme is the same as the
probability of two strings mating, since there is only one pair per
scheme}

 For ToStates := 0 to BigN-1 do begin

  {Set up SumOverB for each mating scheme}
  For MateNum := 0 to BigN-1 do begin
    SumOverB[MateNum] := 0;

    For Crossbit := 0 to l-1 do
      SumOverB[MateNum] := SumOverB[MateNum] +
     IONE[MateNum,Crossbit,ToStates];
  end;{for Matenum}
  {Done Setting up SumoverB}

  For PopSt := 0 to BigN-1 do begin
    BigPSU[Popst,TOStates] := 0;

    For MateNum := 0 to BigN-1 do
      BigPSU[Popst,ToStates] :=
BigPSU[Popst,ToStates] +
MateProb[Popst,Mates[MateNum,0].Iprime,Mates[MateNum,0].Jprime]
*SumOverB[Matenum];

  End; {for Popst}
End; {For ToStates}

{Write out answer}
 Writeln(DataFile);
```

```
 Writeln(DataFile,'Strst',' Action');
 For Strsti := 0 to H do
   Writeln(DataFile,Strsti:5,Action(Strsti):5);
 Writeln(DATAFILE);
 Writeln(DATAFILE, 'Reproduction and crossover');

 Write (DataFile,'   ');
 For ToStates := 0 to BigH-1 do
     Write (DATAFILE,ToStates:5,'  ');

 Writeln(DATAFILE);

 For PopSt := 0 to BigH-1 do begin
   Write (DATAFILE,PopSt:5);

   Sumoverpopst := 0;
   For ToStates := 0 to BigH-1 do begin
     Write (DATAFILE,BigPSU[popSt,ToStates]*100:7:1);
     Sumoverpopst := Sumoverpopst + BigPSU[popSt,ToStates]*100;
   End;
   Writeln(DATAFILE,Sumoverpopst:7:1);
 End;

 For MateNum := 0 to BigH-1 do
   For PrimeNum := 0 to BigH-1 do begin
     {the following four statements are used for clarity}
     Strsti := Mates[MateNum,0].Iprime;
     Strstj := Mates[MateNum,0].Jprime;
     Strstipri := Mates[PrimeNum,0].Iprime;
     Strstjpri := Mates[PrimeNum,0].Jprime;

     Popst := Stateof(strsti,strstj);
     ToStates := Stateof(strstipri,strstjpri);


      If(strsti = strstj) and (strstipri = strstjpri) then
BigPUV[Popst,ToStates] :=
    (Mute[Strsti,Strstipri] * Mute[Strstj,Strstjpri])
      Else
      If (strsti = strstj) and (strstipri < > strstjpri) then
 BigPUV[Popst,ToStates] :=
   2*(Mute[Strsti,Strstipri]*Mute[Strstj,Strstjpri])

      Else
      If (strsti < > strstj) and (strstipri = strstjpri) then
 BigPUV[Popst,ToStates] :=
   (Mute[Strsti,Strstipri]*Mute[Strstj,Strstjpri])

     Else
     If (strsti < > strstj) and (strstipri < > strstjpri) then
BigPUV[Popst,ToStates] :=
  (Mute[Strsti,Strstipri]*Mute[Strstj,Strstjpri]) +
  (Mute[Strsti,Strstjpri]*Mute[Strstj,Strstipri]);
```

```
     End;

   {Write out BigPUV}
    Writeln(DataFile);
    Write (DataFile,'  ');
    For ToStates := 0 to BigN-1 do
Write(DATAFILE,ToStates:5,'  ');

    Writeln(DATAFILE);

    Writeln(DATAFILE,' Mutation');

    For PopSt := 0 to BigN-1 do begin
      Write(DATAFILE,PopSt:5);

      Sumoverpopst := 0;
      For ToStates := 0 to BigN-1 do begin
Write (DATAFILE, BigPUV[popSt,ToStates]*100:7:1);
Sumoverpopst := Sumoverpopst + BigPUV[popSt,ToStates]*100;
      End;

      Writeln(DATAFILE,Sumoverpopst:7:1);
    End;



    Closefiles;
  End.
```

RECENT WORKING PAPERS

1. Albert Marcet and Ramon Marimon
Communication, Commitment and Growth. (June 1991)
[Published in *Journal of Economic Theory* Vol. 58, no. 2, (December 1992)]

2. Antoni Bosch
Economies of Scale, Location, Age and Sex Discrimination in Household Demand. (June 1991)
[Published in *European Economic Review* 35, (1991) 1589-1595]

3. Albert Satorra
Asymptotic Robust Inferences in the Analysis of Mean and Covariance Structures. (June 1991)
[Published in *Sociological Methodology* (1992), pp. 249-278, P.V. Marsden Edt. Basil Blackwell: Oxford & Cambridge, MA]

4. Javier Andrés and Jaume Garcia
Wage Determination in the Spanish Industry. (June 1991)

5. Albert Marcet
Solving Non-Linear Stochastic Models by Parameterizing Expectations: An Application to Asset Pricing with Production. (July 1991)

6. Albert Marcet
Simulation Analysis of Dynamic Stochastic Models: Applications to Theory and Estimation. (November 1991), 2d. version (March 1993)
[Forthcoming in *Advances in Econometrics* invited symposia of the Sixth World Congress of the Econometric Society (Eds. JJ. Laffont i C.A. Sims). Cambridge University Press]

7. Xavier Calsamiglia and Alan Kirman
A Unique Informationally Efficient and Decentralized Mechanism with Fair Outcomes. (November 1991)
[Forthcoming in *Econometrica*]

8. Albert Satorra
The Variance Matrix of Sample Second-order Moments in Multivariate Linear Relations. (January 1992)
[Published in *Statistics & Probability Letters* Vol. 15, no. 1, (1992), pp. 63-69]

9. Teresa Garcia-Milà and Therese J. McGuire
Industrial Mix as a Factor in the Growth and Variability of States'Economies. (January 1992)
[Forthcoming in *Regional Science and Urban Economics*]

10. Walter Garcia-Fontes and Hugo Hopenhayn
Entry Restrictions and the Determination of Quality. (February 1992)

11. Guillem López and Adam Robert Wagstaff
Indicadores de Eficiencia en el Sector Hospitalario. (March 1992)
[Published in *Moneda y Crédito* Vol. 196]

12. Daniel Serra and Charles ReVelle
The PQ-Median Problem: Location and Districting of Hierarchical Facilities. Part I (April 1992)
[Published in *Location Science*, Vol. 1, no. 1 (1993)]

13. Daniel Serra and Charles ReVelle
The PQ-Median Problem: Location and Districting of Hierarchical Facilities. Part II: Heuristic Solution Methods. (April 1992)
[Forthcoming in *Location Science*]

14. Juan Pablo Nicolini
Ruling out Speculative Hyperinflations: a Game Theoretic Approach. (April 1992)

15. Albert Marcet and Thomas J. Sargent
Speed of Convergence of Recursive Least Squares Learning with ARMA Perceptions. (May 1992)
[Forthcoming in *Learning and Rationality in Economics*]

16. Albert Satorra
Multi-Sample Analysis of Moment-Structures: Asymptotic Validity of Inferences Based on Second-Order Moments. (June 1992)
[Forthcoming in *Statistical Modelling and Latent Variables* Elsevier, North Holland. K.Haagen, D.J.Bartholomew and M. Deistler (eds.)]

Special issue Vernon L. Smith
Experimental Methods in Economics. (June 1992)

17. Albert Marcet and David A. Marshall
Convergence of Approximate Model Solutions to Rational Expectation Equilibria Using the Method of Parameterized Expectations.

18. M. Antònia Monés, Rafael Salas and Eva Ventura
Consumption, Real after Tax Interest Rates and Income Innovations. A Panel Data Analysis. (December 1992)

19. Hugo A. Hopenhayn and Ingrid M. Werner
Information, Liquidity and Asset Trading in a Random Matching Game. (February 1993)

20. Daniel Serra
The Coherent Covering Location Problem. (February 1993)

21. Ramon Marimon, Stephen E. Spear and Shyam Sunder
Expectationally-driven Market Volatility: An Experimental Study. (March 1993)
[Forthcoming in *Journal of Economic Theory*]

22. Giorgia Giovannetti, Albert Marcet and Ramon Marimon
Growth, Capital Flows and Enforcement Constaints: The Case of Africa. (March 1993)
[Published in *European Economic Review* 37, pp. 418-425 (1993)]

23. Ramon Marimon
Adaptive Learning, Evolutionary Dynamics and Equilibrium Selection in Games. (March 1993)
[Published in *European Economic Review* 37 (1993)]

24. Ramon Marimon and Ellen McGrattan
On Adaptive Learning in Strategic Games. (March 1993)
[Forthcoming in *A. Kirman and M. Salmon eds.* "Learning and Rationality in Economics" Basil Blackwell]

25. Ramon Marimon and Shyam Sunder
Indeterminacy of Equilibria in a Hyperinflationary World: Experimental Evidence. (March 1993)
[Forthcoming in *Econometrica*]

26. Jaume Garcia and José M. Labeaga
A Cross-Section Model with Zeros: an Application to the Demand for Tobacco. (March 1993)

27. Xavier Freixas
Short Term Credit Versus Account Receivable Financing. (March 1993)

28. Massimo Motta and George Norman
Does Economic Integration cause Foreign Direct Investment?
(March 1993)
[Published in *Working Paper University of Edinburgh* 1993:I]

29. Jeffrey Prisbrey
An Experimental Analysis of Two-Person Reciprocity Games.
(February 1993)
[Published in *Social Science Working Paper 787* (November 1992)]

30. Hugo A. Hopenhayn and Maria E. Muniagurria
Policy Variability and Economic Growth. (February 1993)

31. Eva Ventura Colera
A Note on Measurement Error and Euler Equations: an Alternative to Log-Linear Approximations. (March 1993)

32. Rafael Crespí i Cladera
Protecciones Anti-Opa y Concentración de la Propiedad: el Poder de Voto. (March 1993)

33. Hugo A. Hopenhayn
The Shakeout. (April 1993)

34. Walter Garcia-Fontes
Price Competition in Segmented Industries. (April 1993)

35. Albert Satorra i Brucart
On the Asymptotic Optimality of Alternative Minimum-Distance Estimators in Linear Latent-Variable Models. (February 1993)

36. Teresa Garcia-Milà, Therese J. McGuire and Robert H. Porter
The Effect of Public Capital in State-Level Production Functions Reconsidered. (February 1993)

37. Ramon Marimon and Shyam Sunder
Expectations and Learning Under Alternative Monetary Regimes: an Experimental Approach. (May 1993)

38. José M. Labeaga and Angel López
Tax Silumlations for Spain with a Flexible Demand System. (May 1993)

39. Daniel Serra and Charles ReVelle
Market Capture by Two Competitors: The Pre-Emptive Location Problem. (May 1993)
[Forthcoming in *Journal of Regional Science*]

40. Xavier Cuadras-Morató
Commodity Money in the Presence of Goods of Heterogenous Quality. (July 1993)
[Forthcoming in *Economic Theory*]

41. M. Antònia Monés and Eva Ventura
Saving Decisions and Fiscal Incentives: A Spanish Panel Based Analysis. (July 1993)

42. Wouter J. den Haan and Albert Marcet
Accuracy in Simulations. (September 1993)
[Forthcoming in *Review of Economic Studies*]

43. Jordi Galí
Local Externalities, Convex Adjustment Costs and Sunspot Equilibria. (September 1993)
[Forthcoming in *Journal of Economic Theory*]

44. Jordi Galí
Monopolistic Competition, Endogenous Markups, and Growth. (September 1993)

45. Jordi Galí
Monopolistic Competition, Business Cycles, and the Composition of Aggregate Demand. (October 1993)
[Forthcoming in *Journal of Economic Theory*]

46. Oriol Amat
The Relationship between Tax Regulations and Financial Accounting: a Comparison of Germany, Spain and the United Kingdom. (Nomvember 1993)

47. Diego Rodríguez and Dimitri Vayanos
Decentralization and the Management of Competition. (November 1993)

48. Diego Rodríguez and Thomas M. Stoker
A Regression Test of Semiparametric Index Model Speciication. (November 1993)

49. Oriol Amat and John Blake
Control of the Costs of Quality Management: a Review or Current Practice in Spain. (November 1993)

50. Jeffrey E. Prisbrey
A Bounded Rationality, Evolutionary Model for Behavior in Two Person Reciprocity Games. (November 1993)

51. Lisa Beth Tilis
Economic Applications of Genetic Algorithms as a Markov Process. (November 1993)