

A Behavioral Approach to a Strategic Market Game *

Martin Shubik, *Cowles Foundation, Yale University, and Santa Fe Institute*
Nicolaas J. Vriend, *Queen Mary and Westfield College, University of London*

October 1998

J.E.L. classification codes: C61, C63, C72, C73, D83, D91

Keywords: Market game, dynamic programming, Classifier System, adaptive behavior

M. Shubik, Cowles Foundation for Research in Economics, Yale University, Dept. of Economics, P.O. Box 2125 Yale Station, New Haven, CT 06520-2125, USA.

N.J. Vriend, Queen Mary and Westfield College, University of London, Dept. of Economics, Mile End Road, London, E1 4NS, UK, <n.vriend@qmw.ac.uk>, <http://www.qmw.ac.uk/~ugte173/>.

* We wish to thank Paola Manzini and participants at the Society of Computational Economics conference in Austin, TX, for helpful comments. Stays at the Santa Fe Institute, its hospitality and its stimulating environment are also gratefully acknowledged.

1. Introduction

In this paper we interlink a dynamic programming, a game theory and a behavioral simulation approach to the same problem of economic exchange. We argue that the success of mathematical economics and game theory in the study of the stationary state of a population of microeconomic decision makers has helped to create an unreasonable faith that many economists have placed in models of "rational behavior".

The size and complexity of the strategy sets for even a simple infinite horizon exchange economy are so overwhelmingly large that it is reasonably clear that individuals do not indulge in exhaustive search over even a large subset of the potential strategies. Furthermore unless one restricts the unadorned definition of a noncooperative equilibrium to a special form such as a perfect noncooperative equilibrium, almost any outcome can be enforced as an equilibrium by a sufficiently ingenious selection of strategies. In essence, almost anything goes, unless the concept of what constitutes a satisfactory solution to the game places limits on permitted or expected behavior.

Much of microeconomics has concentrated on equilibrium conditions. General equilibrium theory provides a central example. When one considers infinite horizon models one is faced with the unavoidable task of taking into account how to treat expectations concerning the future state of the system. An aesthetically pleasing, but behaviorally unsatisfactory and empirically doubtful way of handling this problem is to introduce the concept of "rational expectations". Mathematically this boils down to little more than extending the definition of a noncooperative equilibrium in such a way that the system "bites its tail" and time disappears from the model. Stated differently one imposes the requirement that expectations and initial conditions are related in such a manner that the system is stationary. All expectations are self-confirming and consistent. From any two points in time, if the system is in the same physical state overall behavior will be identical.

Unfortunately, even if we were to assume that the property of consistency of expectations were a critical aspect of human life, the noncooperative equilibrium analysis would not tell us how to get there. Even if one knows that an equilibrium exists, suppose that the system is started away from equilibrium, the rational expectations requirement is not sufficient to tell us if it will converge to equilibrium. Furthermore as the equilibrium is generally not unique the dynamics is probably highly influenced by the location of the initial conditions.

The approach adopted here is to select a simple market model where we can prove that for at least a class of expectations formation rules, a unique stationary state exists and we can calculate the actual state. Then we consider what are the requirements to study the dynamics of the system if the initial conditions are such that the system starts at a position away from the equilibrium state.

The model studied provides an example where the existence of a perfect noncooperative equilibrium solution can be established for a general class of games with a continuum of agents.

In the game studied a full process model must be specified. Thus a way of interpreting the actions of the agents even at equilibrium is that equilibrium is sustained by a group of agents where each single agent may be viewed as consisting of a team. One member of the team is a very high IQ problem solver, who on being told by the other member of the team what all future prices are going to be, promptly solves the dynamic program which tells him what to do now, based on the prediction

he has been given. He does not ask the forecaster how he made his forecast. We can, for example, establish the existence of an equilibrium stationary through time based on the simple rule that the forecaster looks at the last price extant in the market and (with a straight face) informs the programmer that that price will prevail forever. But if we do not set the initial conditions in such a way that the distribution of all agents is at equilibrium we do not know *a priori* that the system will actually converge to the equilibrium predicted by the static theory.

An open mathematical question which we do not tackle at this point is how to define the dynamic process and prove that it converges to a stationary equilibrium regardless of the initial conditions of the system. A way of doing this for a specific dynamic process might involve the construction of a Lyapunov function and showing its convergence.

Karatzas, Shubik and Sudderth [1992] (KSS) formulated a simple infinite horizon economic exchange model involving a continuum of agents as a set of parallel dynamic programs and were able to establish the existence of a stationary equilibrium and wealth distribution where individuals use fiat money to buy a commodity in a single market and each obtain a (randomly determined) income from the market. The economic interpretation is that each individual owns some (untraded) land as well as an initial amount of fiat money. Each piece of land produces (randomly) a certain amount of perishable food (or "manna") which is sent to market to be sold. After it has been sold, each individual receives an income which equals the money derived from selling his share. Each individual has a utility function of the form:

$$\sum_0^{\infty} \beta^t \varphi(x_t) , \quad (1)$$

where β is a discount factor, and $\varphi(x_t)$ is the direct utility out of consumption at t . The price of the good each period is determined by the amount of money bid (b) and the amount of good available (q). In particular:

$$p_t = \frac{\sum_{i=1}^n b_t^i}{\sum_{i=1}^n q_t^i} \quad (2)$$

Although in KSS the proof was given for the existence of a unique stationary equilibrium with any continuous concave utility function, in general it is not possible to find a closed form representation of either the optimal policy for each trader or the equilibrium wealth distribution in the society. In an extremely special case, noted below, KSS were able to solve explicitly both for the optimal policy and

the resulting wealth distribution. In two related papers Miller and Shubik [1992] and Bond, Liu and Shubik [1994] considered a simple (nonlearning) simulation and a genetic algorithm simulation of the simple example in KSS and in the latter paper considered also a more complex utility function using linear programming methods to obtain an approximation of the dynamic programming solution in order to compare the performance of the simulation with the solution of the dynamic program.

If our only interest were in equilibrium we could settle for a mathematical existence proof and computational procedures to obtain a specific estimate of the structure of equilibrium when needed. But we know that the infinite horizon consistency check of rational expectations is not merely a poor model of human behavior it tells us nothing about dynamics and it is a method to finesse the real problems of understanding how expectations are formed and how decisions are made in a world with less than super rational game players.

In contrast, approaches such as that of the genetic algorithm of Holland [1992] concentrate on the dynamics of learning. It has been observed that genetic algorithms are not *per se* function optimizers. But even if this is true it is a reasonable question to ask: "If one has a fairly straightforward optimization problem which is low dimensional in the decision variables, where by straight mathematics and computational methods we can at least find a conventional economic solution, what do we get by using a learning simulation approach?" If the learning simulation converges to the formal game theoretic solution then we not only have a dynamics, but also may have a useful simulation device to be used as an alternative to formal computation. If there is a great divergence in results on a simple problem, then we might gain some insights as to why.

2. The Dynamic Programming Approach

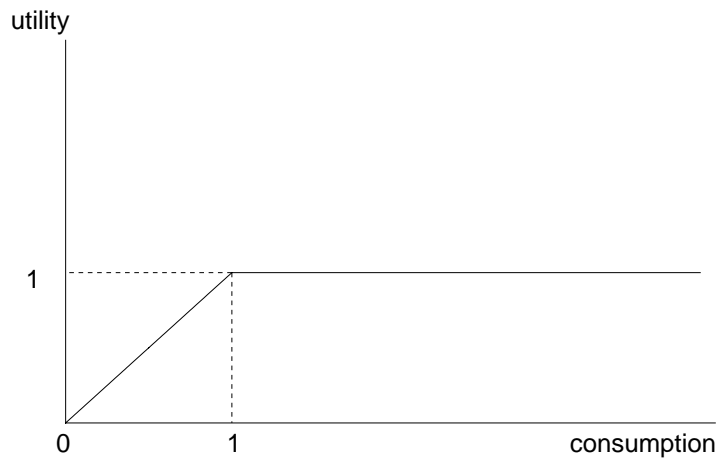
Before we consider the key elements of expectations and disequilibrium and how to approach the adjustment process we confine our remarks to the dynamic programming equilibrium analysis of two special examples which we use as targets for our exploration.

Model 1

We focus the first part of this paper on a simple special case where the utility function for an individual is given by:

$$U(c) = \begin{cases} c; & 0 \leq c \leq 1 \\ 1; & c \geq 1 \end{cases}. \quad (3)$$

The utility function in a single period is illustrated in Figure 1.



A simple utility function

Figure 1

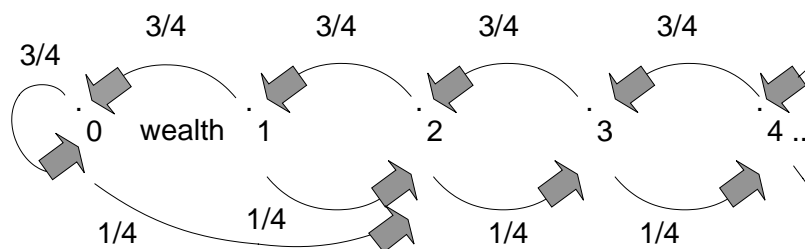
It can be proved that the optimal (stationary) policy of equation (3) has the very simple form

$$c^*(s) = \begin{cases} s; & 0 \leq s \leq 1 \\ 1; & s \geq 1 \end{cases}, \quad (4)$$

where s is the agent's current wealth level. We are able to compute explicitly the value function V as well as the unique invariant measure of the Markov chain when the random distribution of income, represented by y , has the particularly simple form

$$P(y = 2) = \gamma, P(y = 0) = 1 - \gamma \text{ with } 0 < \gamma < \frac{1}{2}. \quad (5)$$

Figure 2 shows the Markov chain, truncated at a wealth level of 4, for $\gamma = 1/4$, where the arrows indicate the transitions between the wealth levels, with the given probabilities.



Markov chain

Figure 2

Suppose that the random variable y has the simple distribution (5). Then the value function $V(\cdot)$ can be computed explicitly on the integers:¹

$$V(0) = A\theta + \frac{\beta}{1-\beta} \quad \text{and} \quad V(s) = A\theta^s + \frac{1}{1-\beta}, \quad s \in \mathbb{N} \quad (6)$$

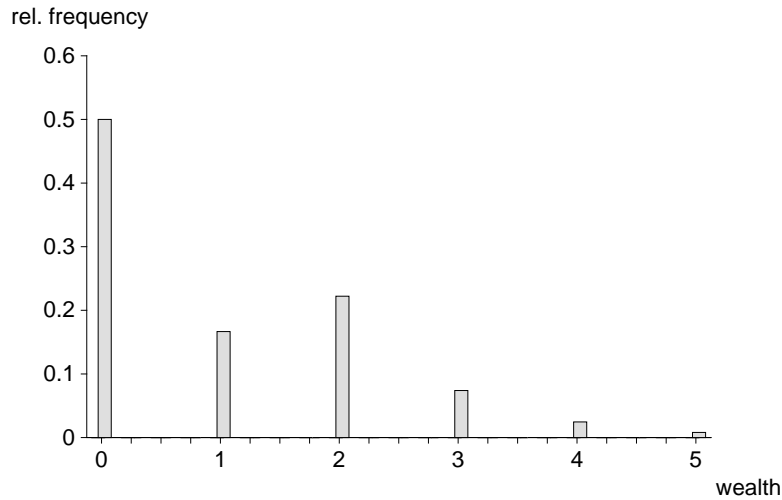
where

$$\theta = \frac{1 - \sqrt{1 - 4\beta^2\gamma(1-\gamma)}}{2\beta\gamma}, \quad A = \frac{1-\gamma}{\gamma\theta\left(1 - \theta + \frac{1-\beta}{\beta\gamma}\right)}. \quad (7)$$

The ergodic Markov chain has an invariant measure $\mu = (\mu_0, \mu_1, \dots)$ given by

$$\mu_0 = c(1-\gamma), \quad \mu_1 = c\gamma, \quad \mu_s = c\left(\frac{\gamma}{1-\gamma}\right)^{s-1} \quad \text{for } s \geq 2, \quad \text{where } c = \frac{1-2\gamma}{1-\gamma}. \quad (8)$$

Suppose for specificity $\gamma = 1/4$ and $\beta = 1/2$, then the stationary wealth distribution is as illustrated in Figure 3.



Wealth distribution
Figure 3

¹ Outside the lattice $V(\cdot)$ is determined by linear interpolation

$$V(s) = \frac{1}{1-p} = \left[\frac{1}{1-\theta} - (s - [s])\theta^{[s]} \right]$$

with $0 \leq s \leq \infty$, where $[s]$ is the integer part of s .

The requirement that individuals use fiat money for bidding is a formalization of the transactions use of money. The motivation to hold money illustrates the precautionary demand to protect the individual in periods of low income.

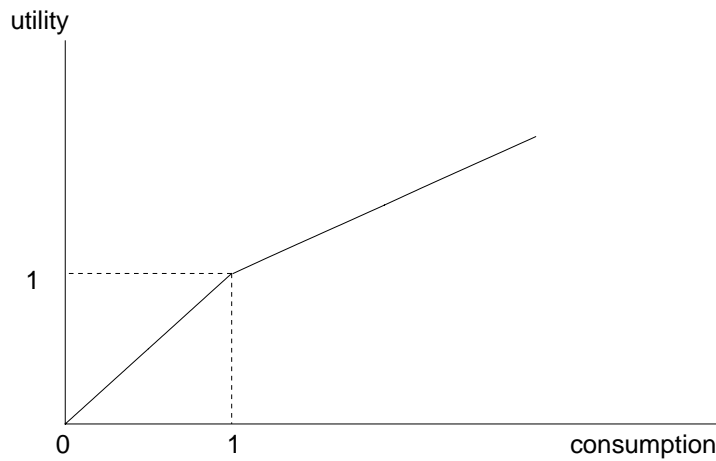
Given the extremely simple form of the optimal policy this example serves as a simple testbed for investigating the basic features of a learning program.

Model 2

We now select an example simple enough to enable us to find an optimal policy and a solution, and yet just rich enough that the pathologies of model 1 are removed. In particular the optimal policy depends directly on β . We suppose that the (one period) utility function is of the form

$$U(c) = \begin{cases} c & ; 0 \leq c \leq 1 \\ 1 + \alpha \cdot (c-1) & ; c \geq 1 \end{cases} \text{ with } 0 \leq \alpha \leq 1 . \quad (9)$$

The utility function is illustrated in Figure 4 for $\alpha=1/2$.



A nonsaturating utility function

Figure 4

For simplicity we consider the probability of gain $\gamma = 1/2$. The Bellman equation for the problem is

$$V(s) = \max_{0 \leq a \leq s} \{ U(a) + \beta/2 [V(s-a) + V(s-a+2)] \} . \quad (10)$$

Let $Q(s)$ be the return function corresponding to $c(s)$. Then Q satisfies

$$Q(s) = \begin{cases} s + \beta/2[Q(0) + Q(2)] , & 0 \leq s \leq 1 \\ 1 + \beta/2[Q(s-1) + Q(s+1)] , & 1 \leq s \leq 2 \\ 1 + \alpha(s-2) + \beta/2[Q(1) + Q(3)] , & s \geq 2 . \end{cases} \quad (11)$$

Notice that, for $1 \leq s \leq 2$, we have $0 \leq s-1 \leq 1$ and $2 \leq s+1$. So

$$Q(s-1) = s - 1 + \beta/2[Q(0) + Q(2)]$$

and

$$Q(s+1) = 1 + \alpha(s-1) + \beta/2[Q(1) + Q(3)] .$$

Also

$$\begin{aligned} Q(0) &= \beta/2[Q(0) + Q(2)] \\ Q(2) &= 1 + \beta/2[Q(1) + Q(3)] . \end{aligned}$$

So we can write the equation for Q as

$$Q(s) = \begin{cases} s + Q(0) & , 0 \leq s \leq 1 \\ 1 + \beta/2[(1+\alpha)(s-1) + Q(0) + Q(2)] , & 1 \leq s \leq 2 \\ 1 + \alpha(s-2) + Q(2) & , s \geq 2 . \end{cases} \quad (12)$$

Differentiate to get

$$Q'(s) = \begin{cases} 1 , & 0 < s < 1 \\ \beta(1+\alpha)/2 , & 1 < s < 2 \\ \alpha , & s > 2 . \end{cases} \quad (13)$$

We can evaluate right and left derivatives by continuity at the end points.

For Q to be concave, we need

$$1 > \frac{\beta(1+\alpha)}{2} \geq \alpha \quad \text{or} \quad \frac{2}{1+\alpha} \geq \beta \geq \frac{2\alpha}{\alpha+1} . \quad (14)$$

This holds for any given β when α is sufficiently close to zero.

To verify the Bellman equation, we will also need that

$$\alpha \geq \beta^2(1+\alpha)/2 \quad \text{or} \quad \frac{2\alpha}{\alpha+1} \geq \beta^2 . \quad (15)$$

It is not difficult to find α and β satisfying all the conditions. For example, take $\beta = 3/4$, $\alpha = 1/2$. Then

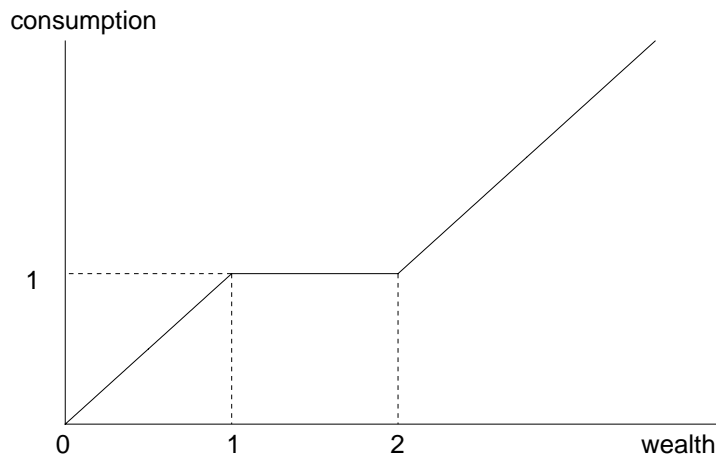
$$\frac{2}{1+\alpha} = \frac{4}{3} > \beta = \frac{3}{4} > \frac{2\alpha}{\alpha+1} = \frac{2}{3} > \beta^2 = \frac{9}{16} . \quad (16)$$

The verification that Q satisfies the Bellman equations is given in a separate paper by Karatzas, Shubik and Sudderth [1995].

For $\beta = 3/4$, and $\alpha = 1/2$, the optimal policy is

$$c(s) = \begin{cases} s, & 0 \leq s \leq 1 \\ 1, & 1 \leq s \leq 2 \\ s-1, & s > 2 . \end{cases} \quad (17)$$

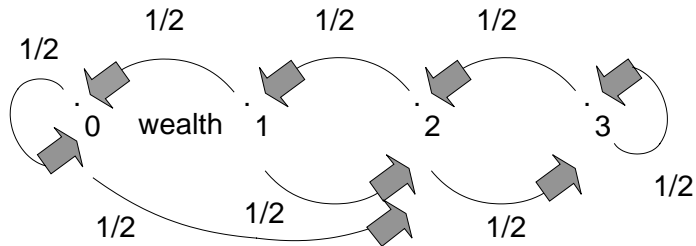
Figure 5 shows the shape of the optimal policy function for these parameter values. We observe that the optimal policy calls for spending all until a wealth level of 1 then saving up to 1 and after saving has reached 1 spending of all further wealth resumes. Furthermore the policy is directly dependent on β .



Optimal policy for the nonsaturating utility function

Figure 5

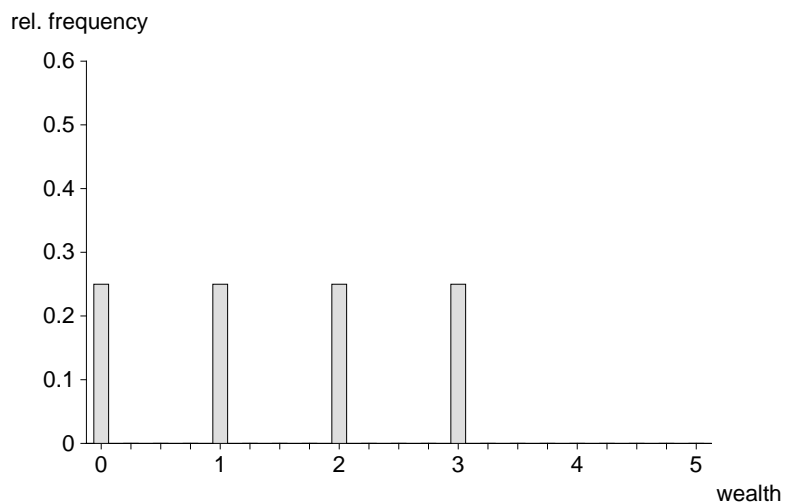
This leads to the Markov chain illustrated in Figure 6, where the arrows indicate the transitions between the wealth levels, with the given probabilities..



Markov chain for the nonsaturating utility function
Figure 6

The stationary wealth distribution is $(\mu_0, \mu_1, \mu_2, \mu_3) = (1/4, 1/4, 1/4, 1/4)$, as presented in Figure 7, and the money supply must be

$$\begin{aligned}
 M &= \frac{1}{4} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{4} \cdot 3 \\
 &= 1\frac{1}{2} .
 \end{aligned}
 \tag{18}$$



Wealth distribution with nonsaturating utility function
Figure 7

For ease in computation and notation we have implicitly assumed a stationary price level of $p = 1$ in the calculations, thus here we note that one unit of the money is bid and $1/2$ a unit remains in hoard.

2.1 Stationary and nonstationary values

The dynamic programming solutions above enable us to calculate stationary values. But if we start away from equilibrium several new problems emerge. The first is : "do we converge to equilibrium?", the second is "how costly is it to get to equilibrium?". It could be that "satisficing" or "good enough" is sufficient. The cost of the search or new routine might be larger than the gain to be had.

2.2 The continuum game and the finite simulation

The mathematical analysis given above was based on results proved for a stochastic economy with a continuum of agents. But the assumption of a continuum of agents is a mathematical convenience to provide mathematical tractability, which must be justified as a reasonable approximation of reality. An open difficult question in game theoretic analysis is does the equilibrium discussed here exist if there is a large finite number of agents, but not a continuum? We conjecture that the answer is yes. The basic functioning of a computer is such that it must deal with a finite number of agents, thus the models we can simulate can represent markets by a large, but finite number of agents. In studying money flows, banking, loan markets and insurance this distinction between large finite numbers and a continuum of agents is manifested in the need for reserves. Reserves play no role in static economic models with a continuum of agents.

3. On Expectations and Equilibrium

In Section 1 we noted that the problem of expectations was finessed in the equilibrium study by imagining that each agent consisted of a team with one individual who could solve dynamic programming problems, while the other agent fed him the information on what expectations he should use to calculate the impact of his policy on future profits. But the key unanswered question is how are these expectations formed. Economic dynamics cannot avoid this question. The need to prescribe an inferential process is not a mere afterthought to be added to an economic model of "rational man", it is critical to the completion of the description of the updating process.

The proof of the existence of an equilibrium supplied by KSS [1992], being based on a complete process model required that the forecasting used in the updating process be completely specified. But as already noted we selected an extremely simple way of forecasting which was to believe that last period's price would last for ever. This clearly contains no learning. If the facts are otherwise our forecaster does not change his prediction.

Any forecasting rule which predicts a constant price in the future and sticks to it regardless of how it has utilized information from the past is consistent with equilibrium. But we are told nothing about whether it converges to the equilibrium, or if it does, then how fast it converges.

Even for the extremely simple economic models provided here, the proof of the convergence of classes of prediction procedures appears to be analytically difficult, although worth attempting. Even were one to succeed, the open problems remain. How do individual economic agents make and revise predictions? Are there any basic principles we can glean concerning inferential processes in economic life? For these reasons we may regard the behavioral simulation approach as a needed complement to and extension of the classical static economic analysis.

4. The Role of Behavioral Simulation and Learning

The past few years has seen an explosion in the growth of computer methods to describe and study learning processes. Among these are Genetic Algorithms (GAs) and Classifier Systems (CSs) (see e.g., Holland [1975], [1986] and [1992], or Machine Learning [1988]). Classifier Systems and Genetic Algorithms are complementary. In combination they are an example of a reinforcement learning algorithm. *“Reinforcement learning is the learning of a mapping from situations to actions so as to maximize a scalar reward or reinforcement signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the highest reward by trying them”* (Sutton [1992], p. 225). A reinforcement algorithm experiments to try new actions, and actions that led in the past to more satisfactory outcomes are more likely to be chosen again in the future. Machine Learning [1992] presents a survey of reinforcement learning.

In the appendix we present the pseudo-code used in our computational analysis, plus a detailed explanation. In this section we focus on the more general ideas underlying the algorithms utilized. A Classifier System (CS) consists of a set of decision rules of the ‘*if ... then ...*’ form. To each of these rules is attached a measure of its strength. Actions are chosen by considering the conditional ‘*if ...*’ part of each rule, and then selecting one or more among the remaining rules, taking into account their strengths. The choice of the rules that will be activated is usually determined by means of some stochastic function of the rules’ strengths.

The virtue of CSs is that it aims at offering a solution to the reinforcement learning or ‘*credit assignment*’ problem. A complex of external payments and mutual transfers of fractions of strengths can be implemented, such that eventually each rule’s strength forms implicitly a prediction of the eventual payoff it will generate when activated. In fact, what CSs do is to associate with each action a strength as a measure of its performance in the past. The essence of reinforcement learning is that actions that led in the past to more satisfactory outcomes, are more likely to be chosen again in the future. This means that the actions must be some monotone function of the (weighted) past payoffs. Labeling these strengths as ‘*predicted payoffs*’ is in a certain sense an interpretation of these strengths, as the CS does not model the prediction of these payoffs, as a process or an act. The basic source from

which these transfers of strengths are made is the external payoff generated by an acting rule. The strengths of rules that have generated good outcomes are credited, while rules having generated bad outcomes are debited. The direct reward from the CS's environment to the acting rule does not necessarily reinforce the right rules. The state in which the CS happens to be may depend, among other things, upon previous decisions. This is important, as only those rules of which the conditional 'if...' part was satisfied could participate in the decision of the current action. Hence, when the current decision turns out to give high payoffs, it may be the rules applied in the past which gave that rule a chance to bid. Moreover in general it may be that not all payoffs are generated immediately, due to the presence of lags or dynamics, implying that the current outcomes are not only determined by the current action, but also partly by some actions chosen previously. This credit assignment problem is dealt with by the so-called '*Bucket Brigade Algorithm*'. In this algorithm each rule winning the right to be active makes a payment to the rule that was active immediately before it. When the CS repeatedly goes through similar situations, this simple passing-on of credit results in the external payoff being distributed appropriately over complicated sequences of acting rules leading to payoff from the environment. Thus the algorithm may '*recognize*' valuable *sequences* of actions.

At the beginning, a CS does not have any information as to what are the most valuable actions. The initial set of rules consists of randomly chosen actions in the agent's search domain, and the initial strengths are equal for all rules.

Given the updated strengths, a CS decides which of the rules is chosen as the current action, where the probability of a rule being activated depends on its strength. This choice of actions is a stochastic function, i.e., it is not simply the strongest rule that is activated, because a CS seeks to balance exploitation and exploration.

A CS is a reinforcement learning algorithm, as experimentation and trying new actions takes place through the stochasticity by which actions are chosen in the CS, and actions that led in the past to more satisfactory outcomes are more likely to be chosen again in the future through the updating of propensities to choose actions. More experimentation takes place when a CS is combined with a Genetic Algorithm (GA), by which new actions can be generated. The frequency at which this is done is determined by the GA rate. Note that a too high GA rate would make that the CS does not get enough time to predict the value of the newly created strings, while a too low GA rate would lead to lack of exploration of new regions.

A GA starts with a set of actions, with to each action attached a measure of its strength. This strength depends upon the outcome or payoff that would be generated by the action. Each action is decoded into a string. Through the application of some genetic operators new actions are created, that replace weak existing ones. GAs are search procedures based on the mechanics of natural selection and natural genetics. The set of actions is analogous to a population of individual creatures, each represented by a chromosome with a certain biological fitness. The basic GA operators are *reproduction*, *crossover* and *mutation*. Reproduction copies individual strings from the old to a new

set according to their strengths, such that actions leading to better outcomes are more likely to be reproduced. Crossover creates a random combination of two actions of the old set into the new one, again taking account of their strengths. This makes that new regions of the action space are searched through. Mutation is mainly intended as a ‘prickle’ every now and then to avoid having the set lock in into a sub-space of the action space. It randomly changes bits of a string, with a low probability.

The key feature of GAs is their ability to exploit accumulating information about an initially unknown search space, in order to bias subsequent search efforts into promising regions, and this although each action in the set refers to only one point in the search space. An explanation of why GAs work is condensed in the so-called ‘*Schema Theorem*’.² When one uses the binary alphabet to decode the actions, then 10110*** would be an example of a ‘*schema*’, where * is a so-called ‘*wild card*’ symbol, i.e., * may represent a 1 as well as a 0. Not all schemata are processed equally usefully, and many of them will be disrupted by the genetic operators; in particular by the crossover operator. The ‘*Schema Theorem*’ says that short, low-order, high performance schemata will have an increasing presence in subsequent generations of the set of actions, where the order of a schema is the number of positions defined in the string, and the length is the distance from the first to last defined position. Although this ‘*implicit parallelism*’ is also sometimes called ‘*randomized parallel search*’, this does not imply directionless search, as the search is guided towards regions of the action space with likely improvement of the outcomes.

GAs are especially appropriate when, for one reason or another, analytical tools are inadequate, and when point-for-point search is unfeasible because of the enormous amount of possibilities to process, which may be aggravated by the occurrence of non-stationarity. But the most attractive feature of GAs is that they do not need a supervisor. That is, no knowledge about the ‘*correct*’ or ‘*target*’ action, or a measure of the distance between the coded actions and the ‘*correct*’ action, is needed in order to adjust the set of coded actions of the GA. The *only* information needed are the outcomes that would be generated by each action. This information is supplied by the CS, which implicitly constructs a prediction of the outcomes for all actions in the set. In this sense GAs exploit the local character of information, and no further knowledge about the underlying outcome generating mechanisms is needed, like e.g., the derivatives of certain functions.

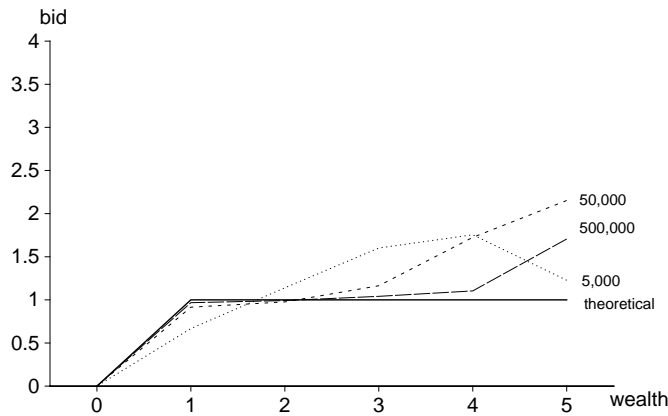
5. Results

We first consider the special case of Model 1, outlined in section 2, with $P(y=2) = 1/4$, $P(y=0) = 3/4$, and $\beta = 0.90$. In order to evaluate the performance of the CS/GA we consider the

² Also called ‘*Fundamental Theorem of Genetic Algorithms*’ (see, e.g., Goldberg [1989] or Vose [1991]).

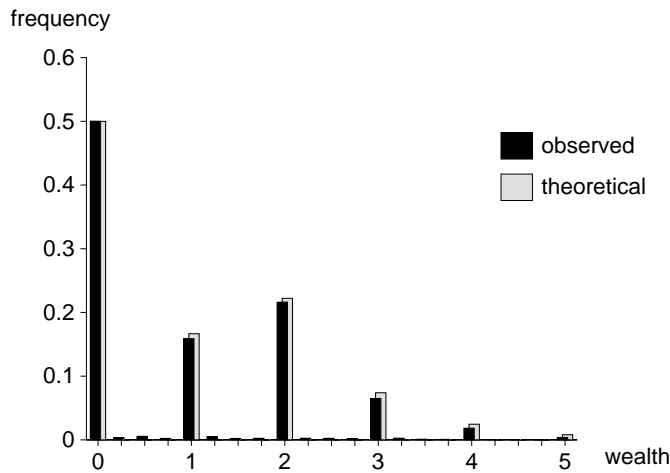
following three measures. First, the actual values of the market bids. Second, the resulting wealth distribution. Third, the average utility realized per period.

Figure 8 presents the bids in absolute value, made at the integer wealth levels ± 0.125 . After 500,000 periods, for the integer wealth levels 1, 2, 3, and 4 the bids come close to the theoretical values. For wealth level 5, bids are coming down towards that level. For the intermediate non-integer wealth levels, not given here, the bids are worse. This is due to the fact that those wealth levels occur much less frequently.



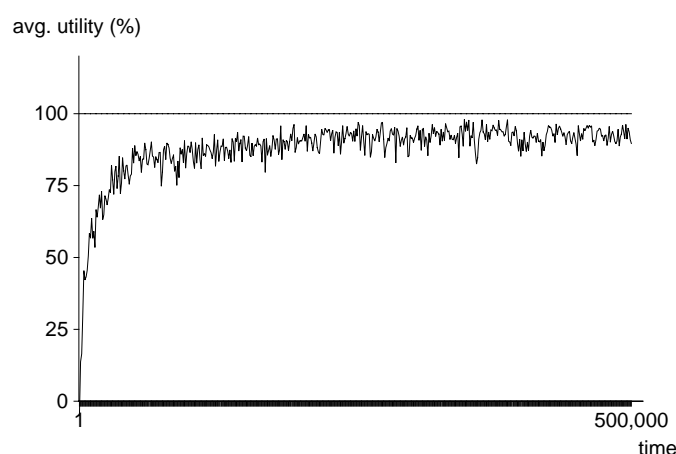
Bids for integer wealth levels: theoretical, and after 5,000, 50,000, and 500,000 periods
Figure 8

Figure 9 shows the resulting wealth distribution, distinguishing the wealth levels 0-0.125, 0.125-0.375, 0.375-0.625, etc. Notice, that the intermediate wealth levels disappear, and that wealth level 5 and higher almost never occur.



Wealth distribution after 500,000 periods
Figure 9

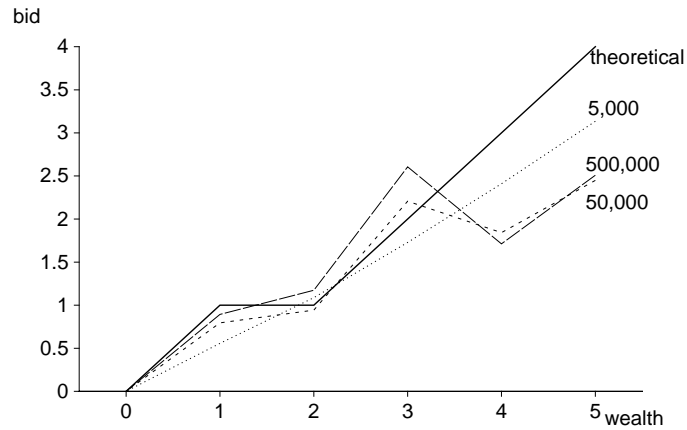
Third, we consider the average utility realized per period. With the optimal policy 50% of the time wealth would be zero, and otherwise wealth is at an integer greater than zero. In the former case utility is zero, while otherwise utility is 1. As a result, the average utility realized per period, following the optimal policy, is 0.50. Considering the last 100,000 periods, our algorithm realized an average utility of 0.49. That would imply an 98% performance level. That does not seem to be adequate, however, since the correct lower bench mark is not zero utility but the utility realized by a zero-intelligent agent. Even random market bids at any wealth level would give an average utility greater than zero. In our case this turns out to be 0.36. Therefore, we normalize the realized utility such that, given the actually realized stochastic income stream, random market bids imply a performance level of 0, and the optimal policy a performance level of 100. Figure 10 gives the whole history; for each observation averaged over 1000 periods.



Average utility realized
Figure 10

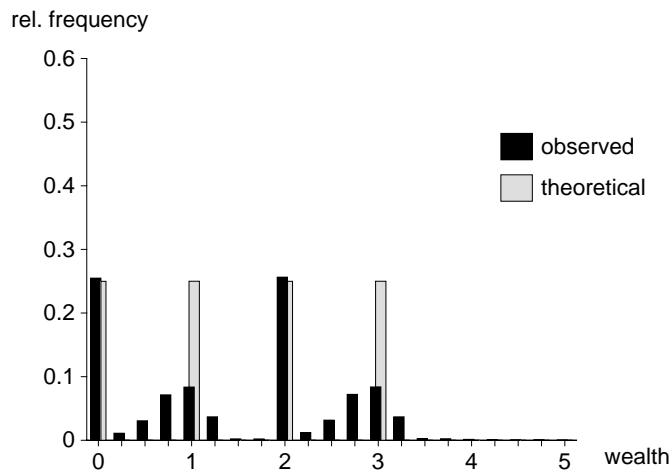
It should be stressed that the algorithm learns only from the actions actually tried by the agent himself. The algorithm can be easily adjusted to incorporate also the following forms of reinforcement learning. First, reinforcement based on the experience of other agents, second, based on hypothetical experience as explained by an advisor, and third, based on virtual experience following the agent's own reasoning process. Including all those reinforcement learning signals would make the algorithm many factors faster, without however, changing the underlying ideas (see also Lin [1992]).

We now turn to model 2 with the nonsaturating kinked utility function, with $\alpha = 0.5$, $P(y=2) = 1/2$, $P(y=0) = 1/2$, and $\beta = 0.75$. Figure 11 shows the bids for the integer wealth levels ± 0.125 .



Bids for integer wealth levels: theoretical, and after 5,000, 50,000, and 500,000 periods
Figure 11

Figure 12 shows the resulting wealth distribution over the wealth levels ± 0.125 for the nonsaturating kinked utility function after 500,000 periods.

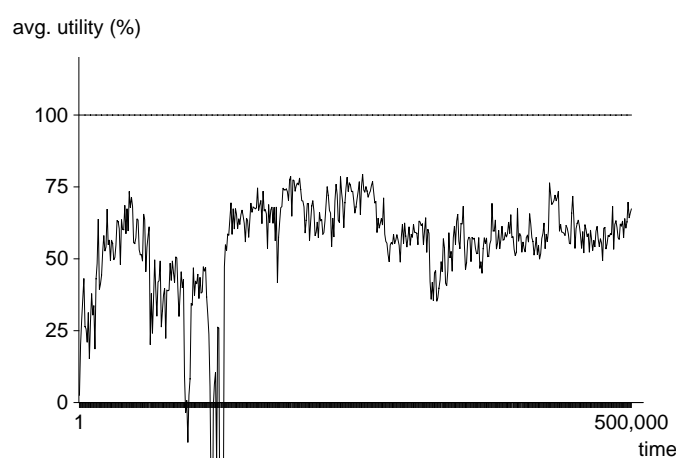


Wealth distribution after 500,000 periods
Figure 12

As we see, the actually learned bid function follows the two kinks in the optimal bid function, but convergence to the latter is not perfect. In particular at wealth levels 2 and 3 consumption is somewhat high. For higher wealth levels consumption is more or less random as these levels almost never occur (see Figure 12), and no learning can take place. The resulting wealth distribution equals the wealth distribution generated by the optimal bids at wealth levels 0 and 2. At wealth levels 1 and 3, the

frequencies are somewhat too low. In fact, the wealth levels just below 1 and 3 occur too often since consumption at wealth levels 2 and 3 is slightly too high.

Figure 13 presents the average utility realized; again normalized to 0 for random behavior and 100 for the optimal strategy. As we see, performance is not as good as for the simple utility function of model 1. There seem to be two reasons for this. First, as shown above, convergence to the optimal consumption levels was not perfect; in particular at wealth levels 2 and 3. Second, the window of opportunity to improve turned out to be much smaller with the nonsaturated utility function in model 2; from 0.80 (random behavior) to 0.87 (optimal choices) instead of the range from 0.36 to 0.50 in model 1.



Average utility realized

Figure 13

6. Conclusion

In economic theory one often makes a distinction between "rational" versus "adaptive" behavior. The first is considered to be "ex ante", "forward looking", and hence "good", whereas the latter is "ex post", "backward looking", and hence "bad". We would argue, however, that this distinction is not so sharp. Every so-called "forward looking" behavior is presuming that some relationships known from the past will remain constant in the future. Moreover as shown in this paper, backward looking agents might learn to behave *as if* they are forward looking. By repeatedly going through similar sequences of actions and outcomes, looking backward, a Classifier System may be able to learn to recognize good sequences of actions, thereby implicitly solving the dynamic programming problem. This has two advantages. First, in cases that an explicit closed form cannot be obtained (e.g., because of less special utility functions), an adaptive algorithm might compute it. Second, we can go on to study economic models with a population of such agents to study dynamics of market economies.

APPENDIX: The Pseudo-Code

```

1  program MAIN;
2  for all agents do                                     { initialization agents }
3  begin
4      with rule[0] do
5          begin
6              action:= 0;
7              fitness:= 0.50;
8          end;
9          for wealth_index:= 1 to 25 do for count:= 1 to 10 do with rule[(wealth_index - 1) * 10 + count] do
10         begin
11             action:= count/10;
12             scale:= 2^length - 1;
13             scaled_action:= round(action * scale);
14             make chromosome of length:= 8 for scaled_action with standard binary encoding;
15             fitness:= 0.50;
16         end;
17         wealth:= 1;
18     end;
19     for period:= 1 to maxperiod do                     { start main loop }
20     begin
21         for all agents do                               { classifier system's actions }
22         begin
23             winning_rule:= AUCTION;                    { see procedure below; lines 53-65 }
24             for all rules do update winning and previously_winning tags;
25             market_bid:= rule[winning_rule].action * wealth;
26             wealth:= wealth - market_bid;
27             with rule[winning_rule] do fitness:= fitness - 0.05 * fitness;
28             bucket:= 0.05 * rule[winning_rule].fitness;
29         end;
30         market_price:= 1;
31         for all agents do                               { classifier system's outcomes }
32         begin
33             consumption:= market_price * market_bid;
34             if consumption<=1 then utility:= consumption else utility:= 1;
35             with rule[winning_rule] do fitness:= fitness + 0.05 * utility;
36             with rule[previously_winning_rule] do fitness:= fitness + 0.9 * bucket;
37             for all rules at the given wealth_index do fitness:= 0.9995 * fitness;
38         end;
39         for all agents do                               { income and wealth change }
40         begin
41             with probability:= 0.25 income:= 2 and with probability:= 0.75 income:= 0;
42             wealth:= wealth + income;
43         end;
44         for all agents do                               { application genetic algorithm }
45         begin
46             for each wealth_index>0 do
47                 if that wealth_index has occurred (100 + ε) times                { with ε ~ N(0, 5) }
48                     since last application of genetic operators at that wealth-index
49                     then GENERATE_NEW_RULE;                                     { see procedure below; 67-82 }
50             end;
51         end;
52     -----

```

```

53 function AUCTION;
54 if wealth=0 then wealth_index:= 0
55     else if wealth<=0.125 then wealth_index:= 1
56     else increment the wealth_index with 1 for each increase in wealth of 0.25;
57 if wealth>5.875 then wealth_index:= 25;
58 if wealth_index=0 then highest_bid:= 0 else for all rules at the given wealth_index do
59 begin
60     linearly rescale fitness such that bid_fitness(max. fitness):= 1 and bid_fitness(avg. fitness):= 0.5;
61     bid:= 0.05 * (bid_fitness + ε);                                     {with ε ≈ N(0, 0.40 → 0.10)}
62     with probability:= 0.10 (→ 0.01) the bid is ignored;
63     determine highest_bid;
64 end;
65 auction:= highest_bid;
66 -----
67 procedure GENERATE_NEW_RULE;
68 choose two mating rules by roulette wheel selection,
69     i.e., each rule drawn randomly with probability:= fitness/sum_fitnesses;
70 with probability:= 0.75 do
71 begin
72     place the two binary strings side by side and choose uniformly random crossing point;
73     swap bits before crossing point;
74     choose one of the two offspring randomly;
75 end;
76 with new_rule do
77 begin
78     fitness:= average fitnesses of the two mating strings;
79     for each bit do with probability:= 0.005 do mutate bit from 1 to 0 or other way round;
80 end;
81 if new_rule is not duplicate of existing rule then replace weakest existing rule with new_rule
82     else re-start GENERATE_NEW_RULE;
83 -----

```

- 1-51 The main program.
- 2-18 Initialization of the agents.
- 9-16 For each of the distinguished wealth levels (see lines 54-57), we create 10 rules. Notice that this differs from Lettau & Uhlig [1996], who model a Classifier System such that every single rule specifies an action for each and every possible state of the world.
- 11 Those rules are placed as a grid with "openings" of size 1/10 on the interval [0, 1].
- 12-14 Since all actions have values in [0, 1], we apply a scaling factor equal to $2^8-1=255$ to the binary strings of length 8. That is, the precision of the possible actions is limited by a term $1/255 \approx 0.004$.
- 15 The initial fitness of all rules is 0.50.
- 17 The initial wealth of each agent is 1.
- 19-51 The main loop of the program.
- 21-29 The agents actions are decided by a Classifier System.
- 23 The winning rule is decided in the AUCTION procedure (see lines 53-65).
- 24 The agent's bid to the market is a fraction of his wealth.
- 27 The winning rule makes a payment of a fraction of 0.05 of it's fitness. The idea is that the rules have to compete for their right to be active. See also below.
- 28 This payment is put in a bucket (see also line 36). See also below.
- 30 The price for the commodity is determined on the market. Here we assume a fixed price.
- 31-38 The agent's Classifier System is updated on the basis of the outcomes obtained.
- 35 The rule gets rewarded from its environment. See also below.
- 36 The contents of the bucket are transferred to the rule that was active the preceding period, discounted at 10%. See also below.
- 37 All rules at the given wealth level pay a small tax from their fitness. See also below.
- 39-43 The agent receives a random income, which changes his wealth.

- 41 With probability 1/4 the agent receives an income of 2, and with probability 3/4 he receives 0.
 44-50 We apply the Genetic Algorithm each $(100+\epsilon)$ times the rules for a given wealth_index have been used, where ϵ is noise with a Normal distribution with expected value 0 and standard deviation 5.
 53-65 The stochastic auction by which the Classifier System decides actions.
 58-65 The winning rule is decided by a stochastic auction.
 60 For the auction, we linearly rescale the fitnesses of all rules.
 61 The variance of the noise in this auction is determined through an annealing schedule; going down from 0.40 to 0.10 over time.
 62 Through a "trembling hand" some experimentation is added. This probability also goes down over time from 0.10 to 0.01.
 67-82 The Genetic Algorithm as such.
 81-82 To prevent complete convergence of the rules, we do not allow for duplicate rules. This is useful in possibly non-stationary environments, where the agent's first interest is not optimizing some fixed objective function, but his capacity to adapt. Moreover, if a rule is very useful, and this is reflected in its fitness, no duplicates are necessary.

27, 28, 35, 36, 37 The dynamic programming problem is:

$$V_t = \max_{c_t} [U(c_t) + \beta \cdot EV_{t+1}] , \text{ where, } V_{t+1} = V(s_t - c_t + y_t) \text{ and } 0 \leq \beta \leq 1$$

The Classifier System with the Bucket Brigade Algorithm works as follows: Each time a rule has been used (lines 23-25), a fraction α is subtracted from its fitness (line 27). This is put in a bucket (line 28), which is used to make a discounted payment to the last rule active right before this rule (line 36). The active rule receives a payoff from its environment (line 35), and will receive a delayed and discounted payment from the bucket filled by the rule that will be active the next time (line 36).

$$\text{Formally: } f_{t+1}^i = f_t^i - \alpha \cdot f_t^i + \alpha \cdot U_t + \beta \cdot \alpha \cdot f_{t+1}^j$$

We can rewrite this as: $(f_{t+1}^i - f_t^i) = \alpha \cdot ((U_t + \beta \cdot f_{t+1}^j) - f_t^i)$ In other words, f^i will increase so long as $(U_t + \beta \cdot f_{t+1}^j) > f_t^i$. Notice the similarity between this Classifier System updating and the dynamic programming problem. A difference with the dynamic programming approach is that instead of taking the value V of an uncertain state, the Classifier System approach assumes that this random value can be represented by the fitness of the rules that happen to be chosen in those states, where this value is given by the direct payoff to those rules plus the value generated by the rules after that. An advantage of the Classifier System approach is that you do not need to know f_{t+1}^j in advance. You just look what is in the bucket one period later. That is, the algorithm is *adaptive*. One additional feature implemented is that we tax all relevant rules at the given wealth level each period (line 37). This makes that you get more easily a distinction between the weak rules that are rarely used and the better rules that get more frequently a payoff.

References

- Machine Learning*. (1988). 3, Nos. 2/3, Special Issue on Genetic Algorithms.
Machine Learning. (1992). 8, Nos. 3/4, Special Issue on Reinforcement Learning.
 Bond, G., Liu, J., & Shubik, M. (1994). *Dynamic Solutions to a Strategic Market Game: Analysis, Programming and a Genetic Algorithm Approach* (mimeo).
 Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.
 Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.
 Holland, J.H. (1986). Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach (Vol. 2)* (pp. 593-623). Los Altos, CA: Morgan Kaufmann.
 Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (2nd ed.). Cambridge, MA: MIT Press.
 Karatzas, I., Shubik, M., & Sudderth, W.D. (1992). *Construction of Stationary Markov Equilibria in a Strategic Market Game* (Working Paper 92-05-022) Santa Fe Institute.

- Karatzas, I., Shubik, M., & Sudderth, W.D. (1995). *A Strategic Market Game with Secured Lending* (Working Paper 95-03-037) Santa Fe Institute.
- Lettau, M., & Uhlig, H. (1996). *Rules of Thumb versus Dynamic Programming* (mimeo).
- Lin, L.J. (1992). Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8, Nos.3/4, 293-321.
- Miller, J.H., & Shubik, M. (1992). *Some Dynamics of a Strategic Market Game with a Large Number of Agents* (Working Paper 92-11-057) Santa Fe Institute.
- Sutton, R.S. (1992). Introduction: The Challenge of Reinforcement Learning. *Machine Learning*, 8, Nos. 3/4, 225-227.
- Vose, M.D. (1991). Generalizing the Notion of Schema in Genetic Algorithms. *Artificial Intelligence*, 50, 385-396.