COWLES FOUNDATION FOR RESEARCH IN ECONOMICS
AT YALE UNIVERSITY

Box 2125, Yale Station
New Haven, Connecticut 06520

COWLES FOUNDATION DISCUSSION PAPER NO. 1144

STOCHASTIC ALGORITHMS FOR DYNAMIC MODELS: MARKOV PERFECT EQUILIBRIUM, AND THE "CURSE" OF DIMENSIONALITY

Ariel Pakes and Paul McGuire

January 1997

Stochastic Algorithms for Dynamic Models: Markov Perfect Equilibrium, and the 'Curse' of Dimensionality.

Ariel Pakes and Paul McGuire *

May 1994 (revised December 1996)

Abstract

This paper provides an algorithm for computing policies for dynamic economic models whose state vectors evolve as ergodic Markov processes. The algorithm codifies a learning process that agents might actually use. It has two features which break the relationship between its computational burden and the dimension of the model's state space. First the integral over future states needed to determine policies is never calculated; rather it is estimated by a simple average of past outcomes. Second, the algorithm never computes policies at all points. Iterations are defined by a location and only policies at that location are computed. Random draws from the distribution determinned by those policies determine the next location. This selection only repeatedly hits the recurrent class of points, a subset of the feasible set whose cardinality is not directly tied to the dimension of the state space. Our motivating example is Markov Perfect Equilibria (a leading model of industry dynamics; see Maskin and Tirole,1988). Though estimators for the primitives of these models are often available, computational problems have made it difficult to examine their implications. We provide numerical results which show that our algorithm can increase speed and decrease memory requirements by several orders of magnitude; opening up new possibilities for applied work.

*Yale University and the NBER, and Yale and the EGC. We thank the referees, G. Gowrisankaran, K. Judd, D. Pollard, J. Rust, and N. Stokey for helpful comments, and the NSF (grant SBR95-12106) for financial support.

Stochastic Algorithms for Dynamic Models: Markov Perfect Equilibrium, and the 'Curse' of Dimensionality.
by Ariel Pakes and Paul McGuire

Applied analysis of dynamic economic models is restricted by the computational problems we run into in attempting to construct optimal, or equilibrium, decision rules. In single agent problems this has become most apparent in the use of estimation algorithms that require the solution of a dynamic programming problem for each trial value of the parameter vector to be estimated. Applied work on multiple agent dynamic equilibrium models has typically been less ambitious. Here the computations are primarily used to obtain the policies implied by parameters estimated elsewhere. These policies are then used to analyze the response of a complex economic system to changes in its environment.[1]

In all cases the policies are typically obtained by computing a fixed point to a mapping of a function whose domain is a subset of $\mathbf{R}^n$, where n is the dimension of the "state space" (the number of relevant state variables) in the problem. In the simplest case this is done by discretization. Each state variable is allowed to take on K distinct values (the grid points), and a fixed point involving K^n points is calculated. Thus without further restrictions the dimension of thisc calculation grows exponentially in n. In multiple agent problems n typically equals the number of state variables per agent times the (maximum) number of agents (ever) active. This is one source of the "curse of dimensionality" in this context. Another is in the computational burden for each agent active at each point. That calculation depends on a probability weighted summation of a function of possible future values of the state variable. The number of possible future values can also grow exponentially in n.

This paper introduces a learning algorithm which can circumvent these two problems for a class of models of interest to economics; models in which the state vector evolves as an ergodic Markov Process. Our algorithm does

[1]Early papers in the single agent tradition include Wolpin,1984, Miller,1984, Pakes,1986, and Rust,1987. Multiagent examples include the I.O. literature on evaluating the impact of different market institutions (eg. Judd, 1992, and Pakes and McGuire,1994), the literature on the implications of macroeconomic policies in a world of heterogeneous agents (eg. Hoppenhayn and Rogerson, 1993), the literature on computing asset prices in financial markets (see the review by Marcet,1994), and the literature on computing the solution to represetative agent growth models (eg. Taylor and Uhlig,1990), and their implications for tax policy (eg. Bizer and Judd,1989).

this by; i) never attempting to obtain accurate policies on the entire state space, and ii) substituting an estimate for the integral over possible future values needed to determine policies.

Recall that in a finite state ergodic Markov Process every sample path will, in finite time, wander into the recurrent class of points, say R, and once in R will stay within it forever (points inside R do not communicate with points outside of it; see, for eg., Freedman,1983, chapter 1). Thus to analyze the impacts of an event from an initial condition in R all we require is knowledge of the policies *on* R. Our algorithm only computes policies for a single point at each iteration, and the process which selects those points eventually confines itself to selecting points in R. Depending on the economics of the problem, the number of points in R need not grow in n at all.

The algorithm uses an average of past draws to estimate the probability weighted summation of the function of future states needed to determine policies. Though this procedure is both faster and has less memory requirements than the alternative of determining policies by explicitly integrating over possible future outcomes, it is also less precise (particularly in early iterations). This generates a tradeoff between the computational burden per point, and the number of iterations needed for a given level of precision. Since the precision of the estimate does not (necessarily) depend on the dimension of the integral being estimated, while the cost of doing the summation explicitly does, the larger is n the more we expect the tradeoff to favor our procedure.

Our algorithm is *asynchronous*, it only calculates policies for a single location at each iteration. Averages of past outcomes are used to form estimates of the expected discounted value (EDV) of the future net cash flows (FNCF) that would result from the different possible actions of each of the agents active at that location. The policy with the highest estimated EDV is chosen as the agent's policy (hence policies are obtained from a simple single agent maximization problem). Jointly, the policies the agents chose determine the distribution of the next iteration's state. A random draw from that distribution is then taken and used to update both; the location of the algorithm, *and* the estimate of the EDV. Use of the draw to both determine the evolution of the state, and to estimate the returns to alternative actions, mimics what would happen were agents actually implementing policies based on our procedure and then using the actual market outcomes to update their estimate of the implications of their actions.

We stress that we do not know of conditions which insure, *a priori*, that our algorithm will converge to the desired policies. However there is a check of whether convergence has occured, and we use it to build a stopping rule for the algorithm[2]. Also, as discussed below, the computational advantages of our algorithm will vary with the structure of the problem (e.g. they should increase in n and be larger when the cardinality of the recurrent class is small relative to that of the state space).

Our numerical results provide an indication of the extent of these advantages when using our algorithm to compute a class of Markov Perfect Equilibria (MPE; see Maskin and Tirole, 1988a and b). They indicate that we can convert problems that would have taken years on the current generation of supercomputers to problems that can be done in a few hours on our work station. Moreover our algorithm seems to be free of convergence problems (which contrasts sharply with previous experience in computing similar MPE).

Difficulties in computing MPE have limited our ability to do applied work in Industrial Organization (I.O.). Though progress has been made in estimating the demand and cost primitives underlying I.O. models, progress that allows us to compute profits and consumer surplus as a function of the distribution of state variables, we have had less success in analyzing how the state variables themselves evolve. This limits our ability to analyze even intermediate run responses to changes in the environment [3]. Since realistic dynamic models for analyzing these responses are far too complicated to solve analytically, further progress will require a mix of estimation and computational strategies: we will have to compute the responses for empirically relevant values of the model's parameters.

This paper proceeds as follows. The next section outlines a Markov Perfect model of industry dynamics. This will both help to focus the subsequent discussion and provide a framework for computing examples. Section 2 considers the burden of computing equilibrium policies for this model using

[2]This is the most we expect from applying fixed point algorithms to functional equations which are not contraction mappings (see, Judd, forthcoming, and the literature cited there). We can prove convergence for a single agent version of our problem (which is a contraction).

[3]For an example of the likely distortions from predictions based on state variables that do not change in response to environmental changes, see Pakes Berry and Levinsohn's (1993) analysis of the response of the average miles per gallon of new car sales to the gas price shock of 1973.

pointwise backward solution algorithms. Section 3 introduces our algorithm and section 4 contains numerical results which provide an indication of its power. A closing section summarizes related computational results. There is an appendix which outlines a computer program which runs our algorithm (and illustrates just how easy it is easy to program).

1 A Simple Model

This section outlines a model of industry dynamics (due to Ericson and Pakes,1995, henceforth EP) that we will work with. The model has firms investing to explore profit opportunities. Successful investments lead to states where the firm earns more profits. Unsuccessful investments, those that leave the firm behind its competitors, lead to a deterioration of profits and may eventually induce owners to exit.

Profits in any period depend on the firm's own level of quality or efficiency, as well as on the levels of efficiency of other competing firms. We let i index levels of efficiency, and assume $i \in \mathcal{Z}^+$, the positive integers. Let $s_i \in \mathcal{Z}^+$ be the number of firms with efficiency level i, so the vector $s = [s_i;\ i \in \mathcal{Z}^+]$ is the "industry structure". Extensions in which i is a vector are discussed below.

Profits for a firm at i when the market structure is s are given by $\pi(i, s)$. We allow $\pi(\cdot)$ to vary with demand and cost primitives, and with the equilibrium assumption. For example, the publically available program for computing our dynamic equilibria contains three examples of $\pi(\cdot)$[4]. The first is a standard differentiated product model where i indexes the quality of a firm's product and equilibrium in the product market is Nash in prices. The second and third examples are both homogeneous product markets in which equilibrium is Nash in quantities. In one i indexes interfirm differences in the marginal cost of production and in the other it indexes interfim differences in capacites. In each case the equilibirum assumption is used to solve for the quantities and prices as a function of state vector, and these are substituted into the profit function to obtain $\pi(i, s)$. [5]

[4]The examples are programed up to a set of parameter values determined by the user. To access a description of, and code for, this algorithm (as well as a number of auxiliary programs designed to help analyze its results); FTP to "econ.yale.edu", use "anonymous" as login, and your own username as "password". Then change directory to "pub/mrkv-eqm" and copy all needed files. There is a "read.me" file to start you off.

[5]Published results that use these examples can be found, respectivley, in: Pakes

Given $\pi(\cdot)$ an incumbent has two choices. It chooses whether to exit or remain active, and if it remains active it chooses an amount of investment. If it exits it receives a selloff value of ϕ dollars (and never reappears). If it invests x it incurs a cost of cx and has a probability distribution of improvements in i which is stochastically increasing in x.

Thus if we let β be the discount rate, and $pr(i', s'|x, i, s)$ provide the firm's perceptions of the joint probability that its own efficiency in the next period will be i' and the industry structure will be s' conditional on (x, i, s), the Bellman equation which determines the EDV of the firm's FNCF $[V(i, s)]$ is given by

$$V(i,s) = \max\{\phi, \pi(i,s) + \sup_{(x \geq 0)} [-cx + \beta \sum V(i',s')pr(i',s'|x,i,s)]\}. \quad (1)$$

The max operator determines if the continuation value of the firm (the expression on the right hand side of ϕ) is greater that the selloff value (ϕ). If so the firm shuts down. If not the firm chooses an amount of investment (an $x \geq 0$) which determines the probability distribution of the increment in the firm's state over the period; i.e. a distribution for

$$i_{t+1} - i_t \equiv \tau_{t+1}.$$

We will assume τ can be written as a difference of two independent random variables, *i.e.*

$$\tau_t \equiv \nu_t - \zeta_t.$$

ν represents the outcome of the firm's investment process and has probabilities given by the family $\mathcal{P} = \{p(\cdot|x), x \in \mathcal{R}^+\}$, which is stochastically increasing in x. ζ is an exogenous random variable with density $\mu(\zeta)$. Its precise interpretation depends on the structure of the profit function but it typically represents common demand or supply conditions (eg.,competition from outside the industry, or factor prices). The distributions of the ν of different firms are independent, but the realization of ζ is common across firms.

---

and McGuire (1994) and Gowrisankaran and Town(1996); EP(1995); and Berry and Pakes(1993) and Gowrisankaran(1995). Note that in all these cases the quantity (or price) choice can be computed without solving the dynamic problem. This is because we have constrained this choice to be a function of (i, s), and the choice does not itself have an independent effect on the evolution of the states. Benkard(1997) extends the program to allow quantity (or price) choices to have an independent effect on the state variables, and Fershtman and Pakes (1997) expand the strategy space to allow for collusive possibilities.

Both ν and ζ are non negative, integer valued, random variables; $\nu = 0$ with probability one if $x = 0$ (a firm cannot advance without some investment), and $\mu(0) > 0$ as is $p(0|x)$ for all finite x.

Thus if $\hat{s_i}$ is the vector providing the states of the competitors of a firm at state i when the industry structure is s, and $q[\hat{s_i}'|i, s, \zeta]$ provides that firm's perceived probability that the states of its competitors in the next period will be $\hat{s_i}'$ conditional on a particular value of ζ

$$pr(i' = i^*, s' = s^*|x, i, s) = \tag{2}$$
$$\Sigma_\zeta p(\nu = i^* \Leftrightarrow i \Leftrightarrow \zeta|x) q[\hat{s_i}' = s^* \Leftrightarrow e(i^*)|i, s, \zeta]\mu(\zeta).$$

where $e(i)$ is a vector which puts one in the i^{th} slot and zero elsewhere (so that $s^* \Leftrightarrow e(i^*)$ lists the states of next years competitors if $s' = s^*$ and $i' = i^*$). Note that $q[\cdot|i, s, \zeta]$ embodies the incumbent's beliefs about entry and exit.

For simplicity we assume there is only one potential entrant a period who pays an amount x_e $(> \beta\phi)$ to enter, and enters one period later at state $\omega^e \in \Omega^e \subset \Omega$ with probability $p^e(\cdot)$. The entrant only enters if the EDV of FNCF from entering is greater than x_e.[6]

We have just described the primitives of the model. We now note the properties of its equilibria that we use below. For regularity conditions which insure these properties see A.1 to A.7 in EP (1995).

EP show that a rational expectations MPE exists for our model. In equilibrium, firm behavior depends on the perceived distributions of future industry structures formalized in the transition probabilities, $q[\hat{s_i}'|i, s, \zeta]$. Yet the investment, entry, and exit choices, generated by that behavior, together with the known distributions of ν given alternative values of x and the distribution of entry locations, generate an objective distribution of industry structures. The model is considered consistent for a given perception if and only if the investment, entry, and exit decisions which result from those perceptions generates an objective distribution of industry structures identical to those perceptions.

EP also show that we will only observe firms at an $i \in \Omega = \{1, ..., K\}$, and that there will never be more than a *finite* number, say $\overline{n}$, of active firms. Thus K is the dimension of the grid, $\overline{n}$ is the number of state variables, and

[6]Different entry models are easy to accommodate provided the distribution of $i's$ at which entry occurs is fixed over time. That is the "ability" of entrants must progress at the same pace as the "ability" of the outside alternative; else entry would eventually go to zero and stay there.

$\Omega \times S$, where $S \equiv \{s = [s_1, ..., s_k] : \sum s_j \leq \overline{n} < \infty\}$, are the possible states, in our analysis. Note that $\#S$ is *finite*, so equilibrium policies can be computed for each $s \in S$.

The heart of the equilibrium is a stochastic process for industry structures [for $\{s_t\}$]. This process is a homogeneous Markov process, *i.e.* if s^t provides the history of industry structures, and $s^t = (s_t, s_{t-1}, ..., s_1)$, then $Pr[s_{t+1} = s'|s^t] = Pr[s_{t+1} = s'|s_t] \equiv Q[s'|s_t]$.

EP also prove that the transition kernel,$Q[\cdot|\cdot]$, associated with each possible equilibrium (and there may be more than one of them) is *ergodic*. Thus there exists a unique positive recurrent class, say $R \subset S$, such that, no matter s_0, s_t will, in finite time, wander into R, and *once in R there is a no probability of communicating outside of R.*

Note, however, that the actual nature of the states in R (e.g. does it include both relatively fractured and relatively concentrated structures?), and the pattern of likely transitions between those states (do we cycle over the divergent types of structures, or are their sudden events that take us more directly from one to another?), depends on the primitives of the model; $\pi(\cdot), \beta, x_e, \phi, \mu(\cdot)$ *and* $\mathcal{P}$. These in turn, depend on demand patterns, technological opportunities, and the institutional structure of the industry; objects that are likely to vary from problem to problem. Thus to use this model for applied work we need some idea of the appropriate primitives and a computation algorithm that allows us to analyze their implications.

2 Equilibrium, the "Backward Solution" Method, and Computational Burden.

Backward solution methods are iterative procedures which start each iteration with a set of numbers in memory, provide a rule which updates those numbers in the course of the iteration, and check to see if the updated numbers satisfy a convergence criteria at the end of the iteration. If not the algorithm begins a new iteration. A detailed description of a backward solution method for our problem is given in Pakes and McGuire (1994; henceforth PM).[7]

[7]For good overviews of backward solution algorithms for dynamic programing problems see Bertsekas,1995, and Judd, forthcoming, chpt.12.

PM's updating procedure is *synchronous*; i.e. it circles through the points in S in a fixed order and updates all estimates associated with *every* $s \in S$ at each iteration. If the values and policies from successive iterations are the same, then the algorithm is said to have converged. Any set of values and their associated policies that are a fixed point to the operator defining the updating rule satisfy the all the equilibrium conditions (conditions 6a to 6d) in EP (1995).

We now modify PM's backwards solution algorithm to make it comparable to the stochastic algorithm introduced below. To do so we need to rewrite the Bellman equation in a way that isolates the integral of future values needed to determine policies. Substituting (2) into (1) and rearranging

$$V(i,s) = \max_{\chi \in \{0,1\}} \{[1 \Leftrightarrow \chi]\phi + \chi \sup_{x \geq 0}[\pi(i,s) \Leftrightarrow cx + \beta \sum_{\nu} w(\nu; i, s)p(\nu|x_1)]\}, \quad (3)$$

where

$$w(\nu; i, s) \equiv \sum_{(\hat{s}_i', \zeta)} V(i + \nu \Leftrightarrow \zeta, \hat{s}_i' + e(i + \nu \Leftrightarrow \zeta))q[\hat{s}_i'|i, s, \zeta]\mu(\zeta). \quad (4)$$

The term, $w(\nu; i, s)$, provides the EDV of FNCF conditional on the current year's investment resulting in a particular value of ν, and the current state being (i, s). It is an expectation because it is constructed by integrating out over the possible outcomes of both the investment strategies of competitors (the $\hat{s}_i'$), and over the outside alternative (the ζ).

Since any set of values for $w(\cdot)$, say $w^+(\cdot)$, determine the values of possible investment outcomes, (3) can use them to determine the investment policy a firm follows if it continues (say x^+). Similarly x^+, $w^+(\cdot)$, and (3) determine a $\chi^+ \in \{0, 1\}$, and this implies a $V^+(\cdot)$. Finaly if, for each possible (i, s), these $V^+(\cdot)$ satisfy (4) when $w^+(\cdot)$ is substituted for the $w(\cdot)$ on the left hand side of that equation, then the equilibrium conditions are satisfied for the policies and value functions generated by $w^+(\cdot)$. Consequently the search for equilibrium policies can be recast as a search for a set of numbers, the $w(\cdot)$, that satisfy the fixed point just described.

Let W be the set of vectors whose elements are the $w(\nu; i, s)$ $((\nu, i, s) \in V \times \Omega \times S)$, and consider the operator $T : W \to W$ defined pointwise as

$$(Tw)(\nu;\ i, s) = \sum_{(\hat{s}_i', \zeta)} V(i + \nu \Leftrightarrow \zeta, \hat{s}_i' + e(i + \nu \Leftrightarrow \zeta)|w)q^w[\hat{s}_i'|i, s, \nu]\mu(\zeta), \quad (5)$$

where

$$V(i,s|w) = \max\{\phi, \pi(i,s) \Leftrightarrow \sup_{x \geq 0}[\Leftrightarrow cx + \beta \sum_{\nu} w(\nu; i, s) p(\nu|x_1)]\},$$

and

$$q^w[\hat{s}_i^{'} = s_i^*|i,s,\nu] \equiv Pr\{\hat{s}_i^{'} = \hat{s}_i^*|i,s,\zeta, \text{ and the policies generated by } \ w\}.$$

A $w \in W$ generates equilibrium policies and value functions *if and only if* it is a fixed point to T. Any such w will be denoted by $w*$[8].

Now consider a backwards solution technique that holds an estimate of w in memory and uses T to update each of its components in each iteration. At each s iteration j begins by computing $V(\cdot|w^{j-1})$ and the associated policies for each incumbent and potential entrant. It then uses these policies to compute a distribution for the future competitors of a firm at (i,s), say $q^{w(j-1)}[\hat{s}_i^{'}|i,s,\zeta]$. Given $V(\cdot|w^{j-1})$ and $q^{w(j-1)}[\cdot|\cdot]$ we obtain $w^j(\cdot;i,s) = (Tw^{j-1})_{(\cdot;i,s)}$ from (5). We stop iterating when $Tw^j \approx w^j$, in which case $w^j = w*$.

The computational burden of such backward solution techniques is essentially the product of three factors, the first two of which generate its "curse" of dimensionality :

- the number of points evaluated at each iteration;
- the time per point evaluated;
- the number of iterations.

Consider the case when there may be more than one state variable per firm. Then $i \in \Omega$ is a "tuple". Let K be the number of distinct tuples in Ω ($K = \#\Omega$; thus if i is a couple whose elements can each take on k values, then $K = k^2$). The rate at which the computational burden of the algorithm increases with an increase in the number of state variables differs depending on whether the increase is in the number of state variables per firm, or in the maximum number of firms ever active (our $\overline{n}$). We focus on the case in which K is fixed and $\overline{n}$ increases. If we have no further restrictions both the number of points evaluated at each iteration, and the time per point evaluated, will

[8]Recall that there may be more than one equilibrium, so we abuse notation slightly by not distinguishing between a particular, and the set of, equilibrium values.

grow exponentially in the number of states per firm. As noted below further restrictions are often available in economic models.

Since each of the $\overline{n}$ active firms can only be at K distinct states, the number of points we need to evaluate at each iteration, or $\#S \leq K^{\overline{n}}$. However symmetry, or more precisely exchangeability, of the value and the policy functions in the state variables of a firm's competitors implies that we do not need to differentiate between two vectors of competitors that are permutations of one another [9]. As shown in Pakes (1993), this insures that an upper bound for $\#S$ is given by the combinatoric $\binom{K+\overline{n}-1}{n}$; but for $\overline{n}$ large enough this bound is tight. The bound increases geometrically (rather than exponentially) in $\overline{n}$ (thus symmetry can be quite helpful). Both our and PM's calculations impose symmetry; so *all* numerical results discussed in this paper use the restrictions implied by symmetry.

The computational burden at a given s, or per point, is primarily determined by the cost of calculating the expected value of future states (of obtaining the $w(\cdot; i, s)$ from the $V(\cdot|w)$ in equation 4 above). Say iteration $j \Leftrightarrow 1$'s policies determine that m firms will be active at point s (accounting for entry and exit). Also assume that there is positive probability on each of κ points for each of the $m \Leftrightarrow 1$ active competitors of a given firm. Then to compute the required expectation we need to sum over κ^m possible future states[10]. Thus the average computational burden per point is proportional to $\sum f(m) m \kappa^m$, where $f(m)$ is the fraction of points with m firms active. The computational burden of obtaining the optimal policies given the evaluation of this summand need not grow in m at all (it does not in the algorithm introduced above).

As an example we consider the calculations in PM (1994). That paper presents an analysis of a differentiated product market in which the products were unidimensional (so each 'i' is an integer), $K = 21$, and $\overline{n}$, the maximum number of firms ever simultaneously active, was 6. If we would have increased market size until $\overline{n} = 10$ then we would have had to compute equilibria with *forty seven* times as many points. Since $\kappa = 2$ the increase of $\overline{n}$ from 6 to

[9] This is the reason that the (i, s) notation for a firm and its competitors is more compact than the more traditional $(i_j, \mathbf{i_{-j}})$ notation. I.e. the more traditional notation cares about the order in which the competitors are listed, while our notation does not.

[10] We sum over a function of each possible future value of the tuple $(\{\nu_j\}_{j=1}^{m-1}, \zeta)$. We could reduce this by using the symmetry restrictions discussed above, but this would require us to find the probabilities associated with each unique $\hat{s}_i'$ vector; a task whose computational burden generally outweighs the gains from using symmetry.

10 would also entail a *twenty two fold* increase in the computational burden per point evaluated. Thus if we optimistically assume both that the number of iterations would not increase when we increased $\overline{n}$, and that there was no memory problems, the increase from $\overline{n} = 6$ to $\overline{n} = 10$ would increase computational time by a factor of over a *thousand*. The typical run in PM took about three hours to run on our work station, so a run with $\overline{n} = 10$ would take over three months, and one with $\overline{n} = 12$ would take several years. Even more telling is that to accomodate two state variables per firm with 21 grid points each and no further restrictions we would have had to increase the number of points evaluated at each iteration by a factor of 1.9×10^8.[11] We come back to this example below.

3 A Stochastic Algorithm.

The stochastic algorithm also iteratively updates an estimate of $w*$. However it combines two distinct ideas to overcome the two aspects of the 'curse' of dimensionality listed above. First it is *asynchronous*, only updating a single location at each iteration, and the selection process which chooses the transitions eventually focuses on the recurrent class of points, or R, a subset of S whose cardinality does not necessarily depend on the dimension of the state space. Second the algorithm reduces the computational burden per point by estimating the integral of future values (the integral in (4)) needed to determine current policies by an average of past monte carlo evaluations of those integrals.

Combining these ideas produces an algorithm that might actually be used by agents attempting to infer optimal behavior from past outcomes. Thus assume $s = s_t$ and that all agents believe that the EDV of FNCF are given by $w(\cdot|\cdot, s_t) = w^+(\cdot|\cdot, s_t)$. They would then choose their policies to maximize the $V(\cdot, s_t|w^+)$ obtained by substituting w^+ for w in (4) above. These choices would generate a distribution of outcomes for each agent's competitors given

[11] With similar issues in mind, Judd (forthcoming) discusses a set of approximation techniques based on fitting parametric functions to only a small fraction of the points in S, and then using the information obtained from those values to predict the value function at other points as needed. We have a separate paper (Pakes and McGuire, 1995) showing how symmetry reduces the dimensionality of these approximations significantly, but our experience with them on problems similar to those considered here was disappointing (we had extreme convergence and accuracy problems). As noted below there is considerable potential for combining approximation techniques with stochastic algorithms.

by $q^{w+}[\cdot|i, s_t]$, and nature would choose the market outcome as a random draw from q^{w+}. The current perception of the *value* of this outcome to firm i is obtained by substituting its $\hat{s}_i$ into the evaluation function given by w^+ and (3). If these values are viewed as random realizations from the integral defining the appropriate components of $w*$, the agents might use them to update their estimates of $w*$. Our algorithm for finding $w*$ mimics the learning algorithm just described.

The j^{th} iteration is defined by its estimate of w, $w^j \in W$, and by its location, $s^j \in S$. Thus the algorithm's updating rule must update both s and w. We begin with the update for s. This requires policies for the incumbents and the potential entrant. The investment and exit policies policies for incumbents, say, $x(\cdot, s_j|w^j)$ and $\chi(\cdot, s_j|w^j)$ are obtained as the solution to

$$\max_{\chi\in\{0,1\}}\{[1 \Leftrightarrow \chi]\phi \; + \; \chi sup_x[\pi(i, s^j) \Leftrightarrow cx + \beta \sum_{\nu} w^j(\nu; i, s^j)p(\nu|x_1)]\}.$$

In the simplest of our entry models the potential entrant pays x_e dollars to enter, and if it enters becomes an incumbent in the next period at location i_e minus the realization of ζ. The entrant enters if the EDV of FNCF from entering exceeds the cost of entry. Thus if we let χ_e equal one if the entrant enters and zero otherwise, the j^{th} iteration's entrant policy is

$$\chi_e(s_j|w^j) = 1 \Leftrightarrow \beta w^j(0; i_e, s^j + e(i_e)) > x_e, \tag{6}$$

where $e(i_e)$ is a K-vector which has one for its i_e element and zero elsewhere.

These policies determine a distribution for s^{j+1}. The actual s^{j+1} is obtained as a random draw from this distribution. To obtain the draw use $\chi(\cdot|w^j)$ to determine which of the firms in s^j remain active, and let the r^{th} active agent's location be i_r^j and its investment be $x_r^j(\cdot|w^j)$. Then for each active agent draw a random variable from the distribution $p(\cdot|x_r^j)$, say ν_r^{j+1}. Also draw ζ^{j+1} from $\mu(\zeta)$. We obtain s^{j+1} as follows. Compute $i_r^j + \nu_r^{j+1} \Leftrightarrow \zeta^{j+1}$ for each active agent. If $\chi_e(w^j) = 1$, also compute $i_e \Leftrightarrow \zeta^{j+1}$ for the potential entrant. Now count how many of these numbers equal i for each $i \in \Omega$. The vector of integers obtained by this procedure is s^{j+1}. Note that if $w^j = w*$ the process generating s^{j+1} would be an ergodic Markov process, and hence would wander into the recurrent class in a finite number of iterations and stay there.

We now update w^j. For each agent and each possible realization of ν, use $V(\cdot|w^j)$ to evaluate the state defined by the actual *simulated draws* for

ζ and for the locations of the agent's competitors; i.e. evaluate

$$V(i+\nu \Leftrightarrow \zeta^{j+1}, \hat{s}_i^{j+1} + e(i+\nu \Leftrightarrow \zeta^{j+1})|w^j).$$

This expression is the j^{th} period evaluation of being in location $(i+\nu \Leftrightarrow \zeta^{j+1})$ when all the other competitors states are determined by their simulated draws. Its expectation conditional on information realized by iteration j is $\sum_{(\hat{s}_i', \zeta)} V(i+\nu \Leftrightarrow \zeta, \hat{s}_i' + e(i+\nu \Leftrightarrow \zeta^{j+1})|w^j) q^{w^j}[\hat{s}_i'|i, s^j, \zeta]\mu(\zeta)$. So if $w^j = w*$, this expectation is $w*$.

Since $V(i+\nu \Leftrightarrow \zeta^{j+1}, \hat{s}_i^{j+1} + e(i+\zeta \Leftrightarrow \zeta^{j+1})|w^j)$ is the current period's perception of the value of a random draw from $w^*(\nu; i, s)$, it is used to update $w^j(\nu; i, s)$. In particular if the random draw is different from $w^j(\nu; i, s)$, then set $w^{j+1} \Leftrightarrow w^j$ equal to a fraction of the difference; i.e. if $\alpha(j, s^j) \in (0, 1)$ set

$$w^{j+1}(\nu; i, s^j) \Leftrightarrow w^j(\nu; i, s^j) = \alpha(j, s^j) \times \tag{7}$$
$$\{V[i+\nu \Leftrightarrow \zeta^{j+1}, \hat{s}_i^{j+1} + e(i+\nu \Leftrightarrow \zeta^{j+1})|w^j] \Leftrightarrow w^j(\nu; i, s^j)\}.$$

Note that if we set $\alpha(j, s^j)$ equal to the inverse of the number of times the estimate of $w^*(\nu; i, s)$ has been updated in the past, then the $w^j(\cdot; i, s)$ are just the sample average of past draws on the EDV of FNCF from (i, s)[12]. Also if $V(i+\nu \Leftrightarrow \zeta^{j+1}, \hat{s}_i^{j+1} + e(i+\nu \Leftrightarrow \zeta^{j+1})|w^j) = w^j(\nu; i, s)$, then $w^{j+1}(\nu; i, s) = w^j(\nu; i, s)$. Consequently if $w^j = w*$ then the expectation of w^{j+1} is $w*$; if we are at $w*$ we will tend to stay their.

We have now shown how to update both w^j and s^j (see the appendix for more detail). We still need an initial estimate of $w*$ and a stopping rule, and the choice of these is determinned by our equilibrium conditions.

3.1 Equilibrium Policies.

We need conditions that define equilibrium policies on a subset of S that is rich enough for subsequent analysis, so we begin by detailing what we mean by "rich enough".

[12]In general the weights need only satisfy Robbins-Monroe type regularity conditions. That is, they must; i)be a function of information available at iteration j, ii) have a sum that tends to infinity as the number of times the point is hit tends to infinity, and iii) have a squarred sum that remains bounded as the number of times the point is hit grows. For a discussion of the importance of these conditions, and of optimal weighting schemes, see Ruppert,1991.

The subvector consisting of the components of w associated with points in $S^* \subset S$ determine the set of entry, exit, and investment policies generated by w for the points in S^* and will be denoted by $w|S^*$. $w|S^*$ allows us to analyze behavior in subgames starting from any point in S^* if, when the policies generated by w are followed, $s_t \in S^*$ implies that the sequence $\{s_\tau\}_{\tau \geq t}$ will be in S^* (with probability one). A $w|S^*$ with this property will be said to generate policies for subgames from S^*.

Definition 1 (Policies for Subgames from S^*) *A $w|S^*$ will be said to generate policies for subgames from S^* if and only if*

$$inf_{\tau \geq t} Pr\{s_\tau \in S^* | s_t, w\} = 1,$$

for each $s_t \in S^*$. •

The next observation shows how to determine whether a given $w|S^*$ generates *equilibrium* policies for subgames from S^*. Notationally if $Q(\cdot, \cdot|w)$ is the Markov transition matrix generated by w and $Q(\cdot, \cdot|w) \equiv [q_{ij}^w]$, then we say that $S^* \to r \ \ [Q(\cdot, \cdot|w)] \ \ iff \ \ q_{ir}^w > 0 \ \ for \ some \ \ i \in S^*$.

Observation 1 *Assume that for a $w \in W$ there is an $S^* \subset S$ such that*

1. *if $S^* \to s' \ [Q(\cdot, \cdot|w)]$, then $s' \in S^*$, and*
2. *$\forall s \in S^*$*
   (a) *if $s_i > 0$ then for each $v \in V$ either*

$$w(\nu; i, s) = \sum_{\hat{s}_i', \zeta} V[i + \nu \Leftrightarrow \zeta, \hat{s}_i' + e(i + \nu \Leftrightarrow \zeta)|w] q^w(\hat{s}_i'|i, s, \zeta) \mu(\zeta).$$

   or the r.h.s. cannot be calculated from $w|S^*$ and $w(\cdot) \geq w^*(\cdot)$,
   (b) *the analogous condition is satsified for $w(0; i^e, s + e(i^e))$, the value assigned to entry at (i, s).*

Then $w|S^*$ generates equilibrium policies (and value functions) for subgames from S^*. •

Condition (1) insures that $Q^{S^*}(\cdot,\cdot|w)$, the matrix formed from the elements of $Q(\cdot,\cdot|w)$ for which $(i,j) \in S^* \times S^*$ defines polices for subgames from S^*. Condition 2 insures that these policies are equilibirum policies. Though any $s \in S^*$ only communicates with other $s \in S^*$ if optimal policies are followed, there are $s \in S^*$ that could communicate with an $s \notin S^*$ for a feasible policy. For these s there is at least one $w(\nu; i, s)$ that is on the boundary of $w|S^*$ in the sense that r.h.s. of (2), or $Tw(\nu; i, s)$, cannot be calculated from $w|S^*$. Equilibrium requires $w = Tw$ if $w(\cdot)$ is not on the boundary and $w(\cdot) \geq w^*(\cdot)$ if it is.

Proof. We begin by showing that $w|S^*$ generates policies for subgames from S^*. Since points in S^* can only transit to other points in S^*, $q^w_{ij} = 0$ whenever $i \in S^*$ but $j \notin S^*$. An inductive argument shows that if Q has this property then so does $Q^\tau \equiv QQ^{\tau-1}$, $\forall \tau \geq 1$. Thus if $s_t \in S^*$, then $Pr\{s_\tau \in S^* \ | s_t, w\} = 1$ $\forall \tau \geq t \ \Rightarrow \lim_{T\to\infty} \inf_{t\leq\tau<T} Pr\{s_\tau \in S^* | s_t, w\} = 1$ $(\forall s_t \in S^*)$. For our policies to be equilibrium policies conditions 6.*a to* 6.*d* in EP (1995) need to be satisfied $\forall s \in S^*$. Simple substitutions show that the equality in (2) and the definitions of $\{Q(\cdot,\cdot|w)\}$ and $\{V(\cdot,\cdot|w)\}$ imply that; the transition probabilities and value functions satisfy (6a) and (6c) $\forall s \in S^*$, and that the policies satisfy 6b and 6d for (i, s) tuples for which the summation on the r.h.s. of (2) can be calculated $(\forall \nu \in V)$. For simplicity assume $V = \{0, 1\}$ (any finite set would do). If $w(0; i, s)$ cannot be calculated, then $\exists \hat{s}_i$ of positive q^w probability with $\hat{s}_i + e(i) \notin S^*$. Since $\mu(\zeta = 0) > 0$, this can only occur if $\chi(i, s|w) = 0 \ \Rightarrow \phi \geq sup_x[\pi(i, s^j) - cx + \beta \sum_\nu w(\nu; i, s^j)p(\nu|x_1)]\} \geq sup_x[\pi(i, s^j) - cx + \beta \sum_\nu w^*(\nu; i, s^j)p(\nu|x_1)]\}$ (from $w(\cdot) \geq w^*(\cdot)$). But then $\chi(i, s|w^*) = 0$ and our policy is optimal. Similarly if $w(1; i, s)$ cannot be calculated but $w(0; i, s)$ can, then $\chi(i, s|w) = 1$ but $0 = x(i, s|w) \geq x(i, s|w^*) \geq 0 \Rightarrow x(i, s|w) = x(i, s|w^*)$. •

We base our initial conditions and stopping rules on Obervation 1 and results from a related aritficial intelligence (AI) literature.

3.2 Lessons from a Related AI Literature.

Go back to the backward solution technique defined by the operator $T : W \to W$ (equation 5). To apply T we need to calculate the sum, $\sum V(\cdot,\cdot|w)q^w(\cdot|\cdot)$, at each point at each iteration. This requires the formula for $q^w(\cdot|\cdot)$ which is both complicated (see equations 6 to 8 of EP, 1995) and puts positive weight on a set of points whose cardinality grows exponentially in the dimension of

the state space.

Since we can obtain random draws from $q^w(\cdot,\cdot)$ without computing the implied probabilities, and it is easy evaluate the $V(\cdot,\cdot|w)$ associated with those draws, the stochastic algorithm substitutes an average of past draws for the needed summation at each point. Were we to do this synchronously, updating all components of w at each iteration, our algorithm would be a direct application of stochastic approximation (Robbins and Monro, 1951). In that literature $w^{j+1} \Leftrightarrow \sum V(\cdot,\cdot|w^j)q^{w(j)}(\cdot|\cdot)$ is a random vector whose conditional distribution (conditional on w^j) may be unknown (in our case it can be constructed from $q^w(\cdot|\cdot)$ but the construction is tedious). The goal is to calculate a root of its conditional expectation (in our case any zero root defines a $w*$).

Blum (1954) provided conditions which insure convergence for the multidimensional case of Robbins and Monroe's problem, and since then the literature has branched out to deal with rates of convergence, asymptotically efficient weighting schemes, etc. in a variety of applied problems (see D.Ruppert, 1991,for an accesible review). Most closely related to our interests is the branch concerned with dynamic programing; the reinforcement (or 'machine') learning literature (see Barto, Bradtke, and Singh, 1995, henceforth BBS, for a review). Since this literature deals with single agent dynamic programming problems, it focuses on cases where the operator T (in equation 5) is a contraction mapping [13].

Much of the machine learning literature is asynchronous; i.e. only the estimates at points associated with the realization of a location variable are backed up at each iteration. Recall that our algorithm is asynchronous with location variable s^j. Early convergence proofs for the asynchronous case required the condition that all points in the state space are recurrent, or $S = R$. The advantage of asynchronous algorithms when $S = R$ results from the fact that in this case the frequency with which a particular point is updated tends to the probability of that point in the ergodic distribution, and the precision of the estimates associated with the point increase in this frequency. Thus provided the estimates that will be used intensively are associated with points with relatively heavy weight in the ergodic distribution, the asynchronous procedure will provide relatively precise estimates of in-

[13]It also focuses on cases where both the states and the controls can take on only a finite set of values, however we have shown that, at least in the synchronous case, convergence does not require the discreteness of the controls.

tensively used points and will not waste much time on points that are rarely used. Extensive numerical analysis indicated that; i) often much of the probability mass in applied problems is concentrated on a very small fraction of the state space, and ii) the imprecision in estimates associatd with points of small probability have little effect on the precision at the points of interest[14].

The intuition underlying the usefulness of the asynchronous algorithms when $R = S$ is even more telling when R is small relative to S (since then we can analyze behavior from any $s \in R$ not knowing policies outside of R), and a part of the AI literature has done away with the $R = S$ requirement for convergence. Among the conditions that replace it is that the inital estimate of the value function be larger than its true value (see the appendix to BBS). This insure that policies which could bring the agents to states that are not in R are tried early on, and only discarded after they are shown to be inoptimal.

Though this literature can guide our choice of initial conditions and stopping rules, it does not contain a set of conditions which insures that our algorithm converges (since our T is not a contraction). On the other hand most of the algorithms in current use for computing fixed points to operators that are not contractions do not insure convergence (see, for eg., Judd, forthcoming). Indeed many of these algorithms cannot determine whether the values they output satisfy the fixed point condition. We now show that though we do not currently have a procedure for determining *a priori* whether our algorithm will converge for a given problem, there is a sense in which we can tell, *a posteori*, whether a given run has produced a w^*.

3.3 Stopping Rules and Initial Conditions.

We still need a stopping rule and an initial condition. To base a stopping rule on a complete check of the conditions in Observation 1 for a given w, we would have to first divide the components of w into those associated with transient and recurrent states, and then split the w associated with recurrent states into those on the boundary and those not. This task is too computationally burdensome to be useful. Instead we use practical approximation procedures for both identifying an S^* for a candidate w, say $S^*(w)$, and determining whether $w|S^*(w)$ generates equilibrium polices.

Recall that $\forall w \in W$ the process generated by $Q[\cdot, \cdot|w]$ is a finite state

[14]In experiments not reported here we found similar results using our algorithm and the examples analyzed in the next section (wherein $S \neq R$ and $T(\cdot)$ is not a contraction).

Markov chain. All such chains have at least one recurrent class (see Freedman,1983, chapter 1), and any recurrent class will satisfy the first condition of observation 1. Thus to obtain $S^*(w)$ we; use $Q[\cdot,\cdot|w]$ to simulate a sequence $\{s_j\}$, let $S^{J_1-J_2}$ be the set of states visited at least once between $j=J_1$ and $j=J_2$, and set $S^*(w)=S^{J_1-J_2}$. Provided both J_1 and $J_2 \Leftrightarrow J_1 \to \infty$, $S^{J_2-J_1}$ will converge to a recurrent class of $Q[\cdot,\cdot|w]$ and satisfy the first condition of observation 1.

To check the equilibrium conditions on this $w|S^*$ we need a norm, a $\|\cdot\|$, to weight deviations from the conditions at different (ν,i,s) tuples. As noted we are less concerned with the precision of our estimates at infrequently visited points, so we use a weighted Euclidean norm with weights equal to the empirical distribution of visits in the simulation run.

Observation 1 requires the norm to check different conditions according as (ν,i,s) is on the boundary of $w|S^*$ or not. To do this we would have to separate out the components of $w|S^*$ on the boundary. The separation program is very time intensive, and since the boundary points tend to be visited *very* infrequently how one treats them has little effect on our norm. Thus our stopping criteria was based on a calculation $\|w \Leftrightarrow Tw\|$ on all of $w|S^*$; when there were attainable future states for which we could not calculate $V(i',s'|w)$, we set their probability equal to zero and renormalized the probabilities of the remainder of the states so their sum equalled one.

Any procedure which uses a norm weighted by the frequency of visits to define a stopping rule (and recall that much of the benefit of the stochastic algorithm stems from focussing disproportionate attention on the more frequently visited states), will face the problem that since boundary points tend to be visited infrequently, their equilibrium conditions will not be given much weight. To ameliorate any biases that this might cause in determining the boundary we employ an initial condition (a w^1) which overestimates $w^*(\cdot)$. If $w^1 > w*$ then all policies that could be optimal will be tried initially, and w^j will tend to approach $w*$ from above (and then *all* equilibirum conditions are satisfied). There are two obvious candidates for such starting values; i)the value function for the one firm problem for the i of the (i,s) couple we are after, and ii)$\pi(i,s)/(1 \Leftrightarrow \beta)$. The value function for the one firm problem is easy to calculate, and $\pi(i,s)$ has to be calculated when we visit the point for the first time anyway (since nothing is in memory for the point then, see the appendix). The relative performance of the two initializations differed with market size (the value function for the one firm problem is not sensitive to the distribution of active competitors whose number increases with market

size). However, even for small market sizes, the profit function initialization typically worked better [15].

Use of a weighted norm for our stopping criteria also implies that we could stop with relatively imprecise estimates at infrequently visited recurrent points. To mitigate any problems that this may generate the output of the algorithm includes a count of how many times each point has been visited. If the analyst needs policies from an infrequently visited s (recurrent or transient) one can either use local restart procedures to obtain more precise estimates in the required neighborhood (see SSB), or revert to pointwise calculations which use the reliably computed values as terminal values for their locations.

There are two important details. First, since outcomes based on later itertion's values are likely to be closer to $w*$ than those from early iterations, the stochastic approximation literature has extensive discussion of efficient weighting schemes (see Ruppert, 1991). These typically downweight early outcomes, a procedure we also found useful. The simple procedure we used restarted the algorithm after a million draws with initial evaluations given by the final values from the prior million, and with $h^j(s) : S \rightarrow Z^+$, the number of times the point s has been visited by iteration j, reset to 1 for those s hit in the last million and to 0 elsewhere. When starting from our initial conditions, it seemed efficient to iterate on this 'averaging' procedure several times before starting up one long run which stopped on our endogenous stopping criteria (though this procedure is likely to be less helpful if w^1 were closer to $w*$).

Second, there is the question of when to perform the test. We stopped the algorithm every million iterations and then calculated firm values for all active firms and the potential entrant at each 's' for which $h^j(s) \geq 1$ twice; once using w^j directly [this produces $V(i, s|w^j)$], and once using the explicit summation $V^*(i, s|w^j) \equiv \Sigma V(i', \hat{s}_i^{'}|w^j) p(i'|i, x, \zeta) q^{w^j}(\hat{s}_i^{'}|i, s, \zeta) \mu(\zeta)$. Our stopping conditions required small values for *both* the weighted means of $V(i, s|w^j) \Leftrightarrow V^*(i, s|w^j)$ and the weighted correlation between these two variables. The weights used for the evaluations at (i, s) were proportional to the number of times the value at (i, s) had been updated since the beginning of our long run [16].

[15] This discussion assumes that we have no additional information on $\{w(\cdot)\}$. If, for example, we had values of $\{w(\cdot)\}$ for a close, though not identical, set of primitives (a situation which will often be the case in applied work), we would make use them. Our point is simply that initial conditons that err on the high side are likely to perform well.

[16] This differs from $h^j(s)$ because there may be more than one firm with the same

Note that the computational burden of the *test* goes up exponentially in the number of state variables, and that this is the *only* part of the algorithm whose burden grows exponentially in the dimension of the state space. As we will see, this is a good reason for switching to a computationally simpler test, at least for high dimensional problems. Tests based on necessary instead of (or in addition to) sufficient conditions, are available (see below)[17].

4 Numerical Results.

The numerical results are meant to indicate the extent to which our algorithm's policies match those generated by $w*$ and its computational burden (including the latter's relationship to the properties of the problem). We answer both questions in the context of the example analyzed in PM(1994). This allows us to compare the results from the stochastic algorithm to the "exact" results used in PM (the primitives in PM's examples were chosen before we knew of the stochastic algorithm, and hence are not related to its properties).

Table 1 compares the investment policies and value functions computed in PM to those obtained from our algorithm. $PS(PE)$ and $VS(VE)$ denote estimates of the probability of $\nu = 1$ conditional on optimal investment,

(i, s) combination. Another modification we used resulted from the fact that, especially in the early iterations, the algorithm tends to pick up points which are hit rarely (if at all) in subsequent iterations. This increases the memory requirements of the algorithm significantly. To circumvent this problem we developed a 'pruning' procedure. After every million draws (i.e. at the same time as the test) the algorithm is instructed to discard (or prune) all those points which were not used since the last pruning. Note also that our testing procedure implicitly assumes that the initial s for the last million iterations is in $R(w)$ and that the w is fairly constant over these iterations; conditons which should be met near the convergence points.

[17]There are several modifications that might increase the efficiency of our algorithm. We have already mentionned the possibility of using alternative weighting schemes and stopping criteria. In addition we have not experimented much with alternative methods for storing and retrieving information. Every time we hit a point we have to retrieve the information we have stored for that point. To store the information we have been using a trinary tree (see the appendix). Consequently the depth of our search grows logarithmically in the number of points stored. There are alternative variants of trees, and alternative storage and retrieval schemes (such as hashing functions) that might prove more efficient. Finaly most of the procedures used in the dynamic programming literature for improving the accuracy of updates could also be used here (see Bertsekas,1995, especially the references to parallel programing, and Judd,forthcoming) .

Table 1: *Exact vs. Stochastic Fixed Point*[1].

	Mean	Standard Deviation	Corr-elaton
PS	.6403	14.46	.9993
PE	.6418	14.43	.....
VS	12.40	643.8	.9998
VE	12.39	639.6	.....

[1]PS(PE) and VS(VE) denote the probabilities and values from the stochastic(exact) algorithms.

and of the value functions, from the stochastic (the exact) algorithm. The means and correlations appearing in the table use the estimates at each (i, s) combination as raw data and then weight them by the number of times the different combinations were visited. The stochastic algorithm used here averaged after each of the first ten million draws, and then ran uninterrupted for an additional ten million.

The correlations between PM's and the stochastic algorithm's estimates of the value functions and of the probabilities were both larger than .999, and the differences between the respective estimates of the means was under .25%. Note that this is in spite of the fact that to ease the computational burden of the 'exact' calculation PM arbitrarily limited the maximum number of firms ever active ($\overline{n}$) to six. Since the stochastic algorithm can increment $\overline{n}$ at little cost it used $\overline{n} = 10$ [18] and found that about .2% of the points in the ergodic distribution had seven or more active firms.

PM uses the output of the exact algorithm and simulation to characterize the dynamic equilibrium. To provide an indication of whether the policies from the stochastic algorithm are close enough for subsequent analysis we ran the *same* simulations using our policies. When we used our simulation to recalculate the tables in PM that are based on continuous valued random variables; for eg. the tables on the *distributions* of the one firm concentration ratio and the average price cost margin, our tables were virtually *identical*

[18]The stochastic algorithm uses the bound on $\overline{n}$ for the procedure which stores information, but never computes values or policies for points at which there are a large number of active firms if those points are not hit by the stochastic process generating s.

to those published in PM.

However, as Table 2 illustrates, there were some differences when we compared the numbers for entry, exit, and the distribution of the number of firms active over periods. These "discrete" dimensions of the equilibrium react discontinuously to *both*; differences in the estimates of the value function between the two algorithms, and to differences in random draws and in the intial s (s_0). We therefore consider whether the differences between column (1) and (2) are due to; differences in s_0 and/or in random draws, or are due to differences in the underlying policies.

Columns 2 to 4 of table 2 are computed from the output from one hundred independent runs of the stochastic algorithm. Each run used the parameter values in PM and was instructed to average over each of the first seven million iterations. Thereafter the algorithm was interrupted every million iterations to run our test and instructed to stop only if the weighted correlation between our test value functions was over .995 and the difference between their weighted means were less than 1% (see the last section). The average (over runs) of the number of iterations at stopping was 5.1 million (after the averaging), but this number ranged from 1 to 26 million.

After satisfying the test criteria each run ran a 10,000 period simulation using the point at which it stopped as an initial condition and its estimate of the $\{w\}$ to determine policies. This was also the length of the simulation run in PM, but all runs differ in s_0 and in random draws.

Columns 2,3 and 4 of table 2 look across the 100 simulation runs and compute; the average, standard deviation, and maxima and minima of the fraction of equilibria with different numbers of firms active. There *is* noticeable variance over runs in these dimensions of the simulation (though not in the *average* number of firms active). However, except possibly at the very upper tail of the distribution, this variance is quite large relative to the differences between columns 1 and 2; and this tail *should* be incomparable since, as noted, in computing the exact fixed point PM assumed that $\overline{n} = 6$.

Column 5 presents the variances in the fraction of equilibria with different amounts of firms active obtained when we held the value and policy function *fixed* and simulated one hundred independent ten thousand iteration runs from it. The figures in column 3 and 5 are very close. Thus almost all the variance in column 3 is due to differences in s_0 and in random draws. Apparently differences in the policies outputted by the stochastic algorithm have little effect on either the discrete or the continuous dimensions of our equilibrium.

Table 2: *"Discrete" Statistics from Simulation Runs.*

	(1)	(2)	(3)	(4)	(5)
		Stochastic Fixed Point			
	Exact Fixed Point	Average 100 runs	Standard Deviation 100 runs	Max/Min 100 runs	Standard Deviation (policies constant)
Percentage of periods with n firms active					
n=					
3	61.9	58.3	02.9	64.0/49.7	02.8
4	34.4	33.7	02.6	40.0/27.7	02.5
5	03.2	06.3	00.8	08.1/04.7	00.7
6	00.5	01.5	00.3	02.4/00.9	00.3
7	00.0	00.2	00.1	00.3/00.0	00.1
8	00.0	00.0	00.0	00.0/00.0	00.0

The market size in PM's simulation was set by their M parameter (it determines the number of consumers being serviced by the market). Table 3 pushes M up from PM's intial $M = 5$ by units of 1 until $M = 10$. The same stochastic algorithm used for table 2 generated the policies inputted for the simulations summarized in this table; but this time we ran the simulation run for 100,000 periods. The bottom panel provides statistics which enable us to assess how the computational burden changes as market size grows. The number of points in the last row refers to the number of points visited at least once in the last million iterations. This will be our approximation to the size of the recurrent class.

There were 21,300 such points when $M = 5$. The maximum number of firms ever active ($\overline{n}$) does increase in M, but the number of points *does not* increase geometrically in $\overline{n}$. The top part of the panel makes it clear why; when we increase M the number of points at which there are a large number of firms active does increase, but the larger market sizes no longer support as many equilibria with a smaller number of active firms (so as we increase M we discard, as well as add, points to R). Indeed though the function relating the number of points to M is initially convex, it then turns concave giving the impression that it may well asymptote to a finite maximum value (Sutton,1991, considers reasons why this might be so).

Table 3: *Comparisons for Increasing Market Size.*

M =	5	6	7	8	9	10
Percentage of equilibria with n firms active						
n=						
3	58.3	00.8	00.0	00.0	00.0	00.0
4	33.7	77.5	48.9	04.4	00.7	00.1
5	06.3	16.8	41.4	62.3	33.0	07.2
6	01.5	04.2	07.3	25.0	44.3	41.8
7	00.2	00.6	02.2	06.5	15.3	34.3
8	00.0	00.1	00.2	01.7	05.9	13.1
9	00.0	00.0	00.0	00.0	00.8	03.5
10	00.0	00.0	00.0	00.0	00.0	00.0
Average n						
	3.43	4.26	4.64	5.39	5.95	6.64
Minutes per Million Iterations						
	5.5	6.5	7.5	8.6	10	11
Minutes per Test						
	3.6	8.15	17.1	42.8	100	120
Number of Iterations (millions)						
	7+5	7+2	7+21	7+4	7+9	7+3
Number of Points (thousands)						
	21.3	30.5	44.2	68.1	98.0	117.5

For memory comparisons we note that when $\overline{n}$=6 a backward solution algorithm that uses all symmetry restrictions requires about 6.4×10^5 points, and if $\overline{n}$ increased to ten it would require about 3.2×10^7 points. That is our algorithm requires only about 3.3% of the points in S when $\overline{n} = 6$, about .4% when $\overline{n} = 10$, and this fraction would undoubtedly decrease further at larger values of $\overline{n}$.

We now consider CPU times (all obtained on a Sun SPARCStation 2). They are determinned by; the time per iteration, the number of iterations, and the test times. The theoretical discussion indicates that the time per iteration should be a function of the distribution of the number of firms active. Thus in our runs the ratio of the average number of active firms to the time per million iterations only increased from 1.53 to 1.65. The number of iterations until our test criteria was satisfied varied quite a bit between runs; but did not increase noticeably in either M or $\overline{n}$. Thus *absent* test times,

the average CPU time needed for our algorithm seems to grow *linearly* (by about 1.65 times) the average number of firms active. Given the theoretical discussion this is as encouraging a result as we could have hoped for.

Recall that the test time in our algorithm should grow as does the time per point in the exact fixed point calculation (exponentially in the number of firms active). As a result the test time in our runs rises from about 3 minutes when $M = 5$ to over two hours when $M = 10$. By $M = 10$ the algorithm spends ten times as much time computing the test statistic after each million iterations as it spends on the million iterations (and this would only get worse if we pushed M up further). This suggests more research on stopping procedure that make less intensive use of our test [19]

Comparing our results to those that used the backward solution technique, we find that when $\overline{n} = 6$ our program took about half the c.p.u. time (about a third if one ignores test time), but when $\overline{n} = 10$ even the most optimistic projection for the backward techniques lead to a ratio of compute time of 1/228 (1/1116 without test times). If $\overline{n}$ grew much beyond that, or if there were more than one state variable per firm, the backward solution technique simply could not be used (even on the most powerful of modern computing equipment) while we have already analyzed such problems with our algorithm (see below).

5 Summary and Related Results.

We provided an algorithm for computing the policies generated by dynamic economic models whose state variables evolve as ergodic Markov processes. The algorithm is *asynchronous*; it only computes policies from a single location at each iteration, and the process which selects points eventually confines itself to the recurrent subset of the state space. Policies at a point are determinned by maximizing an *estimate* of the integral which generates the EDV of alternative actions. The estimate is obtained as a simple average of past outcomes and is easy to calculate. The policies from any point imply a distribution of outcomes, and a random draw from that distribution is used to update both the location of the algorithm and the estimates of the value of the policies. This mimics what would happen if agents were actually

[19] Even procedures that screen on necessary conditions for equilibria before going to the full test can be quite helpful. We have used the fact that if $\{w\}$ is close to $\{w*\}$, then the changes in $\{w\}$ ought to be a martingale to compute such screens.

choosing policies based on our procedures, nature selected a market outcome from the distribution determinned by their actions, and the agents used that outcome to update their estimate of the EDV of alternative actions[20].

Theory indicates that the algorithm's computational advantages should be particularly large for models in which there are a large number of state variable, and the recurrent class is a small subset of the feasible set. There are typically two factors which determine the dimension of the state space in applied I.O. models; market size (which determines $\overline{n}$), and the number of state variables per firm. As indicated by the numerical results, the economics of MP models indicate that a given set of primitives can only support certain configurations of firms in any lasting way. Thus though as market size increases the model will support structures with more active firms, it will no longer support structures where there are a small number of active firms. So as market size increases we both add and subtract points from R and the *net* effect of market size on $\#R$ is not obvious.

It is straightforward to generalize both the model and the algorithm presented here to accommodate the important case in which there is more than one state variable per firm. As noted, without further restrictions $\#S$ grows exponentially in the number of state variables per firm in these models, so even models where $\overline{n}$ is small can quickly become unmanageable[21]. A moments reflection often generates economic reasons for expecting $\#R$ to also grow at a much slower rate than $\#S$ when the number of state variables per firm increases; and we have found this to be the case in the examples we have analyzed. Thus in differentiated product models where the state vector details different characteristics of the products (eg. the size, mpg, and hp of cars), the primitives often indicate that certain combination of characteristics are not demanded at a price greater than their marginal cost (eg. large cars with a high mpg, or small cars with a low mpg). Alternatively consider locational models where there is an initial locational choice and then a plant specific cost of production (or quality of product) which responds to investments. $\#R$ in these models tends to be linear in the number of locations at

[20]Note that though the interpretation of our algorithm as a learning algorithm is a useful pedagogic device, its empirical usefulness depends on several issues. These include the method by which information on past outcomes is made available to the agents currently active, and the number of updates possible per unit of time.

[21]Often further restrictions are available. Models with multiproduct firms, i.e. in which there may be only one state variable per product but there are many products per firm, are an important example (see Gowrisankaran, forthcoming).

which their is entry.

Still as applied work moves to larger and/or more detailed problems we will encounter bigger recurrent classes, and a need for futher computational simplifications. The AI literature for large, single agent, dynamic progaming problems has a set of papers which combine reinforcement learning techniques (similar to those used here) with approximation methods in the spirit of those described in Judd(forthcoming; see SSB,1993, sec. 9 for references in AI); a combination which is potentially quite powerful for the problems we are interested in[22].

The usefulness of our computational techniques will also vary with aspects of the problem not discussed here. For example to compute the equilibria of models in which behavior depends on values "off the equilibrium" path, as is true in many models with collusion, the algorithm will have to to be modified so that it samples from the relevant nonequilibrium paths. We have found, however, that many problems that initially seem ill suited to our techniques can, on deeper reflection, benefit greatly from them, and, partly as a result, we make no attempt to delimit just where the ideas discussed here may be useful.

Finaly, we note that we have *never* had a run of our algorithm that did not converge. This is in *marked contrast* to results using backward solution algorithms where convergence problems do occur and one has to resort to an assortment of costly procedures to overcome them (see PM,1995).

Appendix: Outline of a Computer Program

Let $j \in Z^+$ index the iteration of the algorithm, and s^j provide its location at iteration j. $s^j \equiv (s_1^j, s_2^j, ..., s_k^j)$, so s_i^j is the number of active firms with state vector equal to i at iteration j. $h_s(j) : Z^+ \to Z^+$ denotes the number of times location s has been hit prior to iteration j (note that $h_s(j) = 0$ is possible). Finaly the set of policies (value functions, profits, ...) for each i with $s_i^j > 0$, will be called the permutation cycle of policies (value functions, profits ....) associated with s^j.

We begin with the three subroutines that are called in the algorithm. The algorithm is iterative, so we go next to what is in memory at iteration j, and then show how we update that information in going from iteration j to iteration $j+1$.

[22]Related work on easing the computational burden of single agent estimation problems in economics includes Keane and Wolpin's(1994) use of functional form approximations, and Rust's(1997) use of simulation to approximate the integral determining the implications of alternative choices.

Subroutines Needed.

The first calculates the permutation cylce of profits for a given s, i.e. $\tilde{\pi}(s) = \{\pi(i,s);\ \forall i\ with\ s_i > 0\}$ (see Section 1 for examples). This subroutine is called when a point is hit for the first time, and the profits are stored in memory thereafter.

The second subroutine calculates initial values for the w's associated with an s which is visited for the first time, i.e. it calculates

$$\{\tilde{w}^0(0,s), \tilde{w}^0(1,s)\}$$

As noted in the text, one possibility is to use $\{\pi(i,s)/(1-\beta), \pi(i+1,s)/(1-\beta)\}\ for\ \{w(0;i,s), w(1;i,s)\}$, in which case the initial conditions are taken directly from the profit calculation.

The third subroutine stores and retrieves the information associated with alternative values of s. We used a trinary tree for this purpose. This starts by searching for the location of the active firm with the highest state or i. The search from any given i can move to; i) a larger i, ii) a smaller i, or, having found the correct i, iii) begin a search for the i of the active firm with the next highest i. A "hash" table could be substituted here.

In Memory at Iteration j

For simplicity we assume $\nu \in \{0,1\}$. For each s with $h_s(j) \geq 1$, we have in memory the fourtuple

$$\{\tilde{w}^j(0,s), \tilde{w}^j(1,s), \tilde{\pi}(s),\ and\ h_s(j)\},$$

where a tilde over a variable indicates a permutation cycle of values, ($\tilde{w}^j(1;s) = \{w^j(1;i,s);\ \forall i\ with\ s_i^j > 0\}, \ldots$). No separate information is stored for points which have not been visited (for which $h_s(j) = 0$).

We now assume we are at $s^j = (s_1^j, ..., s_k^j)$, and explain how we generate s^{j+1}.

Step 1: Calculating Policies at s^j.

There are two cases to consider, $h_s(j) \geq 1$ and $h_s(j) = 0$. Let $p(x)$ be the probability that $\nu = 1$.

Assume $h_s(j) \geq 1$. For each $i \in \Omega$ with $s_i^j > 0$ calculate the couple $(\chi, x) \in \{[0,1], R^+\}$ that solves

$$\max(\chi)\{(1-\chi)\phi + \chi sup_x[\pi(i,s^j)$$
$$-x + \beta p(x) w^j(1;i,s^j) + \beta(1-p(x))w^j(0;i,s^j)]\}.$$

I.e the policies are calculated as if our j^{th} iteration's estimates of the EDV of FNCF conditional on the alternative possible realizations of ν are correct. Let these policies be $x^j(i,s^j) = x(i,s^j|w^j)$, and $\chi^j(i,s^j) = \chi(i,s^j|w^j)$.

If $h_s(j) = 0$, call the subroutines that compute profits and initial conditions. Then calculate the policies that maximize the analogous expression with $\{\tilde{w}^0(1;s), \tilde{w}^0(0;s)\}$ substituted for $\{\tilde{w}^j(1;s), \tilde{w}^j(0;s)\}$.

Step 2: Updating $s(j)$.

Recall that a firm's state variable can take on only K values. Begin by setting a set of K counters to zero (these will count the number of firms at each different value of the state variable at iteration $j+1$).

Draw ζ^{j+1}, the change in value of the outside alternative, from $\mu(\zeta)$. Then, starting from the lowest value of i with $s_i^j > 0$, do the following. If $\chi^j(i, s^j) = 0$ skip all firms at s_i^j and go to s_{i+1}^j. If $\chi^j(i, s^j) = 1$, then for each of the s_i^j firms at i draw ν_i^{j+1} from $p[\cdot|x^j(i, s^j)]$, and up the counter at location $i + \nu_i^{j+1} - \zeta^{j+1}$ by one.

This determines exit and the next iterations i for all incumbents that remain active. We need also to account for possible entry. Recall that in the simplest entry model a potential entrant pays x_e dollars to enter, and if it enters becomes an incumbent at location $i = i_e - \zeta^{j+1}$ in the next period. Thus the EDV of FNCF from entering exceeds the cost of entry only if $\beta w^j(0; i_e, s^j + e(i_e)) > x_e$, where $e(i_e)$ is a K-vector which has one for its i_e element and zero elsewhere. If there is entry up the counter at $i_e - \zeta^{j+1}$ by one.

The vector of values from the set of counters (ordered from the lowest location) is s^{j+1}.

Step 3: Updating Memory.

If $h_s(j) = h_s(j+1)$, that is if location s was not hit at iteration j, then the data in memory for location s is not changed. If $h_s(j+1) = h_s(j) + 1$ we do update. Recall that

$$s^{j+1} - e[i + \nu_i^{j+1} - \zeta^{j+1}],$$

is the vector providing the outcomes of the random draws from location s for all but the i^{th} firm. Then for $\nu = \{0, 1\}$

$$w^{j+1}(\nu; i, s) = [h_s(j)/(h_s(j)+1)]\ w^j(\nu; i, s)\ +$$
$$[1/(h_s(j)+1)]\ V^j\{(i + \nu - \zeta^{j+1}, s^{j+1} - e[i + \nu_i^{j+1} - \zeta^{j+1}] + e[i + \nu - \zeta^{j+1}]\}.$$

As noted the couple, $[i + \nu - \zeta_{j+1}, s^{j+1} - e[i + \nu_i^{j+1} - \zeta^{j+1}] + e[i + \nu - \zeta^{j+1}]$, provides the state that would have been achieved had the firm's own research outcome been ν, but those of its competitors and of the outside alternative been determined by the realizations of the simulated random variables from Step 2. Thus the $(j+1)^{st}$ estimate of the expected discounted value of FNCF conditional on the research outcome ν is a weighted average of; i) the iteration j estimate of that value [with weight equal to $h_s(j)/(h_s(j)+1)$], and ii) the j^{th} iteration's evaluation of the state determined by $i + \nu - \zeta(j+1)$, and the *actual realization* of the simulated research outcomes of all competitors.

If $h_s(j) = 0$ we use $\{w^0(0; i, s), w^0(1; i, s)\}$ for $\{w^j(0; i, s), w^j(1; i, s)\}$ and do the same calculation.

That completes the iteration. Return to step 1 and continue. •

References.

Barto, AG, SI Bradtke and S Singh (1995); "Learning to Act Using Real-Time Dynamic Programming", *Artificial Intelligence*, Vol 72, pp 81-138.

Berry, S. and A. Pakes (1993), "Some Applications and Limitations of Recent Advances in Empirical Industrial Organization: Merger Analysis," the *American Economic Review*, Papers and Proceedings, Vol. 83, pp247-52.

Bizer, D. and K. Judd (1989), "Taxation and Uncertainty", the *American Economic Review* Papers and Proceedings, pp331-336.

Blum J. (1954): "Multivariate Stochastic Approximation Methods", *Annals of Mathematics and Statistics*, 25, 37-751.

Ericson R., and A.Pakes (1995); "Markov Perfect Industry Dynamics: A Framework for Empirical Work", the *Review of Economic Studies*, 62, pp.53-82.

Freedman, D.(1983), *Markov Chains*, Springer Verlag, New York.

Fershtman, C., and A.Pakes (1997); "Simple Dynamic Games with Collusive Possibilities", *mimeo*, Yale University.

Gowrisankaran, G. (1995), "A Dynamic Analysis of Mergers", unpublished *Ph.D. Dissertation*, Yale University.

Gowrisankaran, G. (forthcoming), "Efficient Representation of State Spaces for Some Dynamic Models", *Journal of Economic Dynamics and Control.*

Gowrisankaran G. and R. Town (1996), "Dynamic Equilibrium in the Hospital Industry", forthcoming, *The Journal of Economics and Management Strategy* (special issue on Health Care).

Hopenhayn,H., and Rogerson, R.(1993), "Job Turnover and Policy Analysis: A General Equilibrium Analysis", *Journal of Political Economy*, 101, pp.915-938.

Judd,K. (forthcoming), *Numerical Methods in Economics*, M.I.T. press, Cambridge, Mass.

Judd,K. (1992), "Cournot vs. Bertrand: A Dynamic Resolution", *mimeo.*, the Hoover Institution, Stanford Ca.

Keane, M. and K. Wolpin (1994); "The Solution and Estimation of Discrete Choice Dynamic Programming Models by Simulation and Interpolation: Monte Carlo Evidence", *Review of Economics and Statistics*, 76-4,pp.648-72.

Maskin, E. and J. Tirole (1988a and b) "A Theory of Dynamic Oligopoly: I and II" *Econometrica*, Vol.56, pp.549-99.

Marcet, A. (1994), "Simulation Analysis of Dynamic Stochastic Models", in J.J.Laffont and C.Sims (ed.s), *Advances in Econometrics, Proceedings of the Sixth World Congress of the Econometric Society*, Cambridge University Press, N.Y.

Miller, R.(1984) "Job Matching and Occupational Choice", *Journal of Political Economy*, 92, pp.1086-1120.

Pakes, A. (1986), "Patents as Options: Some Estimates of the Value of Holding European Patent Stocks", *Econometrica*, pp.755-84.

Pakes, A. (1992), "Dynamic Structural Models, Problems and Prospects", in J.J.Laffont and C.Sims (ed.s), *Advances in Econometrics, Proceedings of the Sixth World Congress of the Econometric Society*, Cambridge University Press, N.Y.

Pakes,A., S. Berry, and J.Levinsohn (1993), "Applications and Limitation of Some Recent Advances in I.O.: Price Indexes and the Analysis of Environmental Change", *American Economic Review*, Papers and Proceedings, 83, pp 240-46.

Pakes A., and P.McGuire (1994), "Computing Markov-Perfect Nash Equilibria: Numerical Implications of a Dynamic Differentiated Product Model, the *RAND Journal of Economics*, 25 No.4., pp 555-589.

Pakes A., and P.McGuire (1995), "Computing Markov-Perfect Nash Equilibria II: Approximations", *mimeo.*, Yale University.

Robbins, H., and S. Monroe, (1951), "A Stochastic Approximation Method", *Annals of Mathematics and Statistics*, 22, pp.400-407.

Ruppert, D. (1991); "Stochastic Approximation", in the *Handbook of Sequential Analysis*, B.Ghosh and P.K. Sen (ed.s), Marcel Dekker Inc. , New York.

Rust J. (1987),"Optimal Replacement of GMC Bus Engines:An Empirical Model of Harold Zurcher", *Econometrica*,55, pp999-1034.

Rust J. (1997), "Using Randomnization to Break the Curse of Dimensionality" *Econometrica*,65,pp487-516.

Sutton, J. (1991), *Sunk Costs and Market Structure*, M.I.T. Press.

Taylor, J. and H. Uhlig (ed.s),"Solving Nonlinear Stochastic Growth Models: A Comparison of Alternative Solution Methods", the *Journal of Economic And Business Statistics* 8, pp 1-55.

Wolpin, K.(1994); "An Estimable Dynamic Model of Fertility and Child Mortality", the *Journal of Political Economy*,92, 852-874.