Dynamic Programming

entry for consideration by the

New Palgrave Dictionary of Economics

John Rust, *University of Maryland*[*]

April 5, 2006

[*]Department of Economics, University of Maryland, 3105 Tydings Hall, College Park, MD 20742, phone: (301) 404-3489, e-mail: `jrust@gemini.econ.umd.edu`. This draft has benefited from helpful feedback from Kenneth Arrow, Daniel Benjamin, Larry Blume, Moshy Buchinsky, Larry Epstein, Chris Phelan.

1 Introduction

Dynamic Programming is a recursive method for solving *sequential decision problems* (hereafter abbreviated as SDP). Also known as *backward induction,* it is used to find *optimal decision rules* in "games against nature" and *subgame perfect equilibria* of dynamic multi-agent games, and competitive equilibria in dynamic economic models. Dynamic programming has enabled economists to formulate and solve a huge variety of problems involving sequential decision making under uncertainty, and as a result it is now widely regarded as the single most important tool in economics. Section 2 provides a brief history of dynamic programming. Section 3 discusses some of the main theoretical results underlying dynamic programming, and its relation to game theory and optimal control theory. Section 4 provides a brief survey on numerical dynamic programming. Section 5 surveys the experimental and econometric literature that uses dynamic programming to construct empirical models economic behavior.

2 History

A number of different researchers in economics and statistics appear to have independently discovered backward induction as a way to solve SDPs involving risk/uncertainty in in the mid 1940s. von Neumann and Morgenstern (1944) in their seminal work on game theory, used backward induction to find what we now call *subgame perfect equilibria of extensive form games*.[1] Abraham Wald, the person credited with the invention of *statistical decision theory,* extended this theory to sequential decision making in his 1947 book *Sequential Analysis*. Wald generalized the problem of gambler's ruin from probability theory and introduced the *sequential probability ratio test* that minimizes the expected number of observations in a sequential generalization of the classical hypothesis test. However the role of backward induction is less obvious in Wald's work. It was more clearly elucidated in the 1949 paper by Arrow, Blackwell and Girshick. They studied a generalized version of the statistical decision problem and formulated and solved it in a way that is a readily recognizable application of modern dynamic programming. Following Wald, they characterized the optimal rule for making a statistical decision (e.g. accept or reject a hypothesis), accounting for the costs of collecting additional observations. In the section, "The Best Truncated Procedure" they show how the optimal rule can be approximated "Among all sequential procedures not requiring more than N observations ..." and solve for the optimal truncated sampling procedure "by induction backwards" (p. 217).

Other early applications of backward induction include the work of Pierre Massé (1944) on statistical hydrology and the management of reservoirs, and Arrow, Harris, and Marschak's (1951) analysis of optimal inventory policy. Richard Bellman is widely credited with recognizing the common structure underlying SDPs, and showing how backward induction can be applied to solve a huge class of SDPs under uncertainty. Most of Bellman's work in this area was done at the RAND Corporation, starting in 1949. It was there that he invented the term *dynamic programming* that is now the generally accepted synonym for backward induction.[2]

[1]"We proceed to discuss the game Γ by starting with the last move $\mathcal{M}_\nu$ and then going backward from there through the moves $\mathcal{M}_{\nu-1}, \mathcal{M}_{\nu-2} \cdots$." (p. 126)

[2]Bellman, (1984) p. 159 explained that he invented the name "dynamic programming" to hide the fact that he was doing mathematical research at RAND under a Secretary of Defense who "had a pathological fear and hatred of the term, research." He settled on "dynamic programming" because it would be difficult give it a "pejorative meaning" and because "It was something not even a Congressman could object to."

3 Theory

Dynamic programming can be used to solve for optimal strategies and equilibria of a wide class of SDPs and multiplayer games. The method can be applied both in discrete time and continuous time settings. The value of dynamic programming is that it is a "practical" (i.e. *constructive*) method for finding solutions to extremely complicated problems. However continuous time problems involve technicalities that I wish to avoid in this short survey. If a continuous time problem does not admit a closed-form solution, the most commonly used numerical approach is to solve an approximate discrete time version of the problem or game, since under very general conditions one can find a sequence of discrete time DP problems whose solutions converge to the continuous time solution the time interval between successive decisions tends to zero (Kushner, 1990).

I start by describing how dynamic programming is used to solve single agent "games against nature." I then show how it can be extended to solve multiplayer games, dynamic contracts, and principal-agent problems, and competitive equilibria of dynamic economic models. I discuss the limits to dynamic programming, particularly the issue of *dynamic inconsistency* and other situations where dynamic programming will not find the correct solution to the problem.

3.1 Sequential Decision Problems

There are two key variables in any dynamic programming problem: a *state variable* s_t, and a *decision variable* d_t (the decision is often called a "control variable" in the engineering literature). These variables can be vectors in R^n, but in some cases they might be *infinite-dimensional* objects.[3] The state variable evolves randomly over time, but the agent's decisions can affect its evolution. The agent has a *utility* or *payoff function* $U(s_1,d_1,\dots,s_T,d_T)$ that depends on the realized states and decisions from period $t=1$ to the *horizon* T.[4] Most economic applications presume a *discounted, time-separable* objective function, i.e. U has the form

$$U(s_1,d_1,\dots,s_T,d_T)=\sum_{t=1}^{T}\beta^t u_t(s_t,d_t) \tag{1}$$

where β is known as a *discount factor* that is typically presumed to be in the $(0,1)$ interval, and $u_t(s_t,d_t)$ is the agent's *period t utility (payoff) function*. Discounted utility and profits are typical examples of time separable payoff functions studied in economics. However the method of dynamic programming does not require time separability, and so I will describe it without imposing this restriction.

We model the uncertainty underlying the decision problem via a family of history and decision-dependent conditional probabilities $\{p_t(s_t|H_{t-1})\}$ where $H_{t-1}=(s_1,d_1,\dots,s_{t-1},d_{t-1})$, denotes the *history* i.e. the realized states and decisions from the initial date $t=1$ to date $t-1$.[5] This implies that in the most general case, $\{s_t,d_t\}$ evolves as a history dependent stochastic process. Continuing the "game against nature" analogy, it will be helpful to think of $\{p_t(s_t|H_{t-1})\}$ as constituting a "mixed strategy" played by "Nature" and the agent's optimal strategy is their *best response* to Nature.

[3]In Bayesian decision problems, one of the state variables might be a *posterior distribution* for some unknown quantity θ. In general, this posterior distribution lives in an infinite dimensional space of all probability distributions on θ. In heterogeneous agent equilibrium problems state variables can also be distributions: I will discuss several examples in section 3.

[4]In some cases $T=\infty$, and we say the problem is *infinite horizon*. In other cases, such as a life-cycle decision problem, T might be a random variable, representing a consumer's date of death. As we will see, dynamic programming can be adapted to handle either of these possibilities.

[5]Note that this includes all deterministic SDPs as a special case where the transition probabilities p_t are degenerate. In this case we can represent the "law of motion" for the state variables by deterministic functions $s_{t+1}=f_t(s_t,d_t)$.

The final item we need to specify is the *timing of decisions*. Assume that the agent can select d_t *after* observing s_t, which is "drawn" from the distribution $p_t(s_t|H_{t-1})$.[6] The agent's choice of d_t is restricted to a *state dependent constraint (choice) set* $D_t(H_{t-1},s_t)$. We can think of D_t as the generalization of a "budget set" in standard static consumer theory. The choice set could be a finite set, in which case we refer to the problem as *discrete choice,* or D_t could be a subset of R^k with non-empty interior, then we have a *continuous choice* problem. In many cases, there is a mixture of types of choices, which we refer to as *discrete-continuous choice problems*.[7]

Definition: A (single agent) *sequential decision problem* (SDP) consists of 1) a *utility function* U, 2) a sequence of *choice sets* $\{D_t\}$, and 3) a sequence of *transition probabilities* $\{p_t(s_t|H_{t-1})\}$ where we assume that the process is initialized at some given initial state s_1.

In order to solve this problem, we have to make assumptions about how the decision maker evaluates alternative risky strategies. The standard assumption is that the decision maker maximizes *expected utility*. I assume this initially and subsequently discuss whether dynamic programming applies to *non-expected utility maximizers* in section 3.6. As the name implies, an expected utility maximizer makes decisions that maximize their *ex ante* expected utility. However since information unfolds over time, it is generally not optimal to *precommit* to any fixed sequence of actions $(d_1,\dots,d_T)$. Instead, the decision maker can generally obtain higher expected utility by adopting a *history-dependent strategy* or *decision rule* $(\delta_1,\dots,\delta_T)$. This is a sequence of *functions* such that for each time t the realized decision is a function of all available information.[8] Under our timing assumptions the information available at time t is (H_{t-1},s_t), so we can write $d_t = \delta_t(H_{t-1},s_t)$.[9] A decision rule is *feasible* if it also satisfies $\delta_t(H_{t-1},s_t) \in D_t(H_{t-1},s_t)$ for all (s_t,H_{t-1}). Each feasible decision rule can be regarded as a "lottery" whose payoffs are utilities, the expected value of which corresponds to expected utility associated with the decision rule. An *optimal decision rule* $\delta^* \equiv (\delta_1^*,\dots,\delta_T^*)$ is simply a feasible decision rule that maximizes the decision maker's expected utility

$$\delta^* = \underset{\delta\in\mathcal{F}}{argmax}\, E\left\{U\left(\{\tilde{s}_t,\tilde{d}_t\}_\delta\right)\right\}, \tag{2}$$

where $\mathcal{F}$ denotes the class of feasible history-dependent decision rules, and $\{\tilde{s}_t,\tilde{d}_t\}_\delta$ denotes the stochastic process induced by the decision rule $\delta \equiv (\delta_1,\dots,\delta_T)$. Problem (2) can be regarded as a static, *ex ante* version of the agent's problem. In game theory, (2) is referred to as the *normal form* or the *strategic form* of a dynamic game, since the dynamics are suppressed and the problem has the superficial appearance of a static optimization problem or game in which an agent's problem is to choose a best response, either to nature (in the case of single agent decision problems), or other rational opponents (in the case of games). The strategic formulation of the agent's problem is quite difficult to solve since the solution is a *sequence of history-dependent functions* $\delta^* = (\delta_1^*,\dots,\delta_T^*)$ for which standard constrained optimization techniques (e.g. the *Kuhn-Tucker Theorem*) are inapplicable.

[6]The alternative case where d_t is chosen before s_t is realized requires a small change in the formulation of the problem.

[7]An example is commodity price speculation, see e.g. Hall and Rust (2005), where a speculator has a discrete choice of whether or not to order to replenish their inventory and a continuous decision of how much of the commodity to order. Another example is retirement: a person has discrete decision of whether to retire and a continuous decision of how much to consume.

[8]In the engineering literature, a decision rule that does not depend on evolving information is referred to as an *open loop* strategy, whereas one that does is referred to as a *closed-loop* strategy. In deterministic control problems, the closed-loop and open-loop strategies are the same since both are simple functions of time. However in stochastic control problems, open-loop strategies are a strict subset of closed-loop strategies.

[9]By convention we set $H_0 = \emptyset$ so that the available information for making the initial decision is just s_1.

3.2 Solving Sequential Decision Problems by Backward Induction

To carry out backward induction, we start at the last period, T, and for each possible combination (H_{T-1}, s_T) we calculate the time T *value function* and *decision rule*.[10]

$$\begin{aligned} V_T(H_{T-1}, s_T) &= \max_{d_T \in D_T(H_{T-1}, s_T)} U(H_{T-1}, s_T, d_T) \\ \delta_T(H_{T-1}, s_T) &= \underset{d_T \in D_T(H_{T-1}, s_T)}{argmax} U(H_{T-1}, s_T, d_T), \end{aligned} \tag{3}$$

where we have written $U(H_{T-1}, s_T, d_T)$ instead of $U(s_1, d_1, \ldots, s_T, d_T)$ since $H_{T-1} = (s_1, d_1, \ldots, s_{T-1}, d_{T-1})$. Next we move backward one time period to time $T-1$ and compute

$$\begin{aligned} V_{T-1}(H_{T-2}, s_{T-1}) &= \max_{d_{T-1} \in D_{T-1}(H_{T-2}, s_{T-1})} E\{V_T(H_{T-2}, s_{T-1}, d_{T-1}, \tilde{s}_T)|H_{T-2}, s_{T-1}, d_{T-1}\} \\ &= \max_{d_{T-1} \in D_{T-1}(H_{T-2}, s_{T-1})} \int V_T(H_{T-2}, s_{T-1}, d_{T-1}, s_T) p_T(s_T|H_{T-2}, s_{T-1}, d_{T-1}) \\ \delta_{T-1}(H_{T-2}, s_{T-1}) &= \underset{d_{T-1} \in D_{T-1}(H_{T-2}, s_{T-1})}{argmax} E\{V_T(H_{T-2}, s_{T-1}, d_{T-1}, \tilde{s}_T)|H_{T-2}, s_{T-1}, d_{T-1}\}, \end{aligned} \tag{4}$$

where the integral in equation (4) is the formula for the conditional expectation of V_T, where the expectation is taken with respect to the random variable $\tilde{s}_T$ whose value is not known as of time $T-1$. We continue the backward induction recursively for time periods $T-2, T-3, \ldots$ until we reach time period $t = 1$. The equation for the value function V_t in an arbitrary period t is defined recursively by an equation that is now commonly called the *Bellman equation*

$$\begin{aligned} V_t(H_{t-t}, s_t) &= \max_{d_t \in D_t(H_{t-1}, s_t)} E\{V_{t+1}(H_{t-1}, s_t, d_t, \tilde{s}_{t+1})|H_{t-1}, s_t, d_t\} \\ &= \max_{d_t \in D_t(H_{t-1}, s_t)} \int V_{t+1}(H_{t-1}, s_t, d_t, s_{t+1}) p_{t+1}(s_{t+1}|H_{t-1}, s_t, d_t). \end{aligned} \tag{5}$$

The decision rule δ_t is defined by the value of d_t that attains the maximum in the Bellman equation for each possible value of (H_{t-1}, s_t)

$$\delta_t(H_{t-t}, s_t) = \underset{d_t \in D_t(H_{t-1}, s_t)}{argmax} E\{V_{t+1}(H_{t-1}, s_t, d_t, \tilde{s}_{t+1})|H_{t-1}, s_t, d_t\}. \tag{6}$$

Backward induction ends when we reach the first period, in which case, as we will now show, the function $V_1(s_1)$ provides the expected value of an optimal policy, starting in state s_1 implied by the recursively constructed sequence of decision rules $\delta = (\delta_1, \ldots, \delta_T)$.

3.3 The Principle of Optimality

The key idea underlying why backward induction produces an optimal decision rule is called

The Principle of Optimality: *An optimal decision rule $\delta^* = (\delta_1^*, \ldots, \delta_T^*)$ has the property that given any $t \in \{1, \ldots, T\}$ and any history H_{t-1} in the support of the controlled process $\{s_t, d_t\}_{\delta^*}$, δ^* remains optimal for the "subgame" starting at time t and history H_{t-1}. That is, δ^* maximizes the "continuation payoff" given by the conditional expectation of utility from period t to T, given history H_{t-1}:*

$$\delta^* = \underset{\delta}{argmax} E\{U(\{s_t, d_t\}_\delta) | H_{t-1}\}. \tag{7}$$

[10] We will discuss how backward induction can be extended to cases where T is random or where $T = \infty$ shortly.

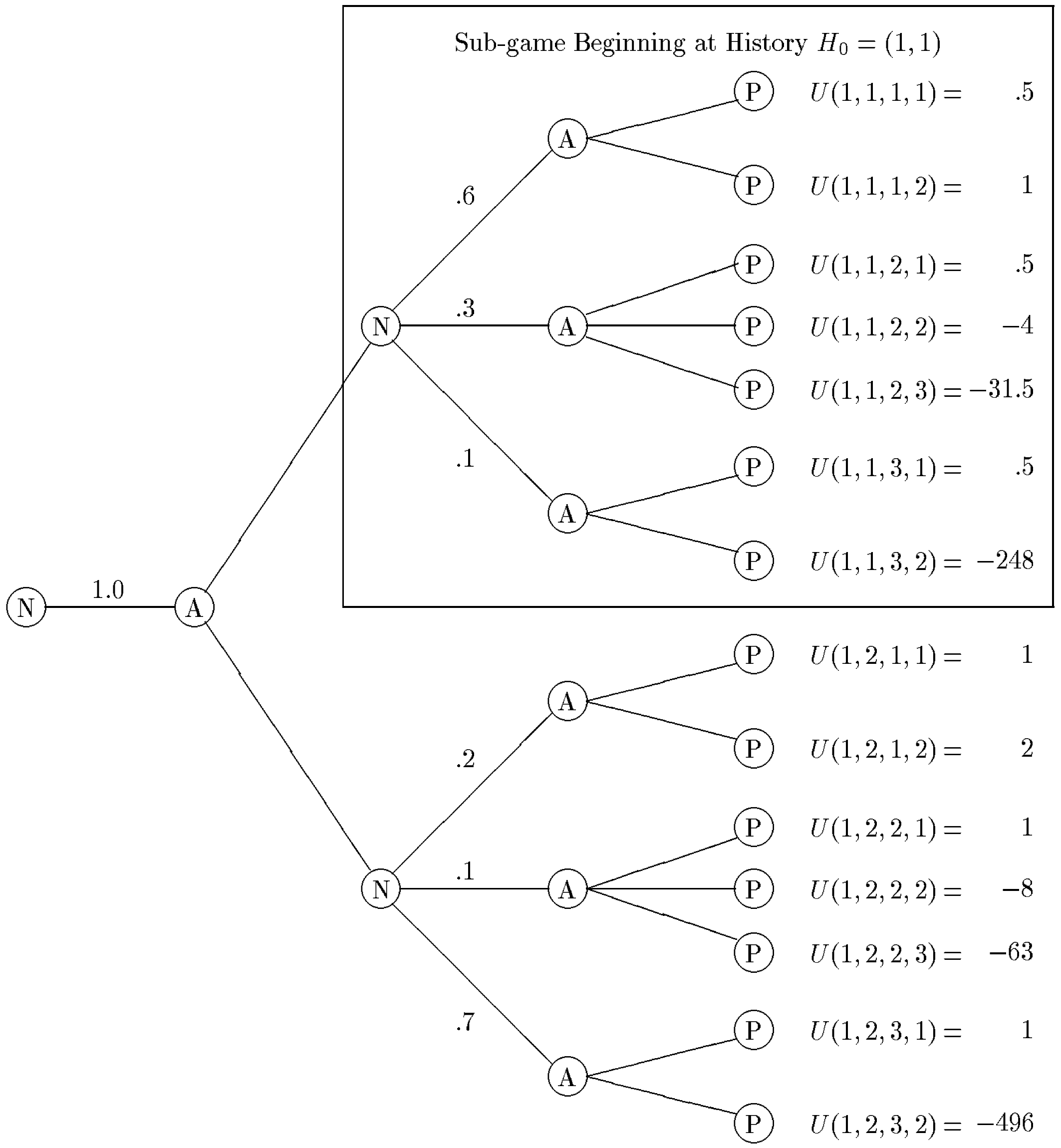

Figure 1: Example Decision Tree

In game theory, the principal of optimality is equivalent to the concept of a *subgame perfect equilibrium* in an *extensive form game*. A when all actions and states are discrete, the stochastic decision problem can be diagrammed as a *game tree* as in figure 1. The nodes marked with "N" represent the possible moves by Nature, whereas the nodes marked "A" are the agent's possible actions. The line segments from the N nodes represent the probabilities by which nature takes various actions. At the end of the decision tree are the final payoffs, the realized utilities. I order the branches of the game tree consecutively, so that action 1 denotes the highest branch, action 2 the next highest and so forth. Thus, a history of the game is equivalent to following a particular path through the tree. For example, the history $H_2 = (1,1,1,1)$ denotes the top most path where the process starts in state $s_1 = 1$, the agent takes decision $d_1 = 1$, nature chooses state $s_2 = 1$, and the agent takes a final action $d_2 = 1$. The payoff to this history is $U(1,1,1,1) = .5$. The subtree outlined in a rectangular box is the *subgame* emanating from the node $H_1 = (1,1)$. This subtree is also a stochastic decision problem. The principle of optimality (or in games, the concept of a subgame perfect equilibrium), guarantees that if δ^* is an optimal strategy (or equilibrium strategy) for the overall game tree, then it must also be an optimal strategy for every subgame, or more precisely, *all subgames that are reached with positive probability from the initial node*.

I illustrate backward induction by solving the decision problem in figure 1. In the final period, $T = 2$, there is no further uncertainty, so we simply solve a collection of *static, deterministic* optimization problems, one for each possible value of (H_{T-1}, s_T). In figure 1 there are 6 possible values of (H_1, s_2), and these are the 6 'A' nodes at the end of the decision tree. At each of these nodes the agent simply chooses the action that results in the highest terminal utility. Thus for the top most 'A' node in figure 1, corresponding to outcome $(H_1, s_2) = (1,1,1)$, we have $V_2(1,1,1) = 1$ and $\delta_2(1,1,1) = 2$, i.e. the optimal decision is to choose the lower branch ($d_2 = 2$) since its utility, $U(1,1,1,2) = 1$ is higher than the upper branch ($d_2 = 1$), which yields a utility value of $U(1,1,1,1) = .5$. Once we solve each of these problems, we compute the expected values for each of the two "N" nodes. In terms of our notation, the expected payoff to choosing either branch is $E\{V_2(s_1,d_1,\tilde{s}_2)|s_1,d_1\}$ which equals

$$\begin{aligned} E\{V_2(s_1,d_1,\tilde{s}_2)|s_1,d_1)\} &= \int V_2(s_1,d_1,s_1)p_2(s_1|s_1,d_1) \\ &= 0.6\times 1.0 + 0.3\times 0.5 + 0.1\times 0.5 = 0.80 \quad \text{when } d_1 = 1 \\ &= 0.2\times 2.0 + 0.1\times 1.0 + 0.7\times 0.1 = 1.06 \quad \text{when } d_1 = 2. \end{aligned} \tag{8}$$

It follows that the optimal decision at $t = 1$ is $\delta_1(s_1) = 2$ and this choice results in an expected payoff of $V_1(s_1) = 1.06$. Dynamic programming has found the optimal strategy (δ_1, δ_2) in 2 steps, after computation of 7 optimizations and 2 conditional expectations.

It should now be evident why there is a need for the qualification "for all H_{t-1} in the support of $\{s_t, d_t\}_{\delta^*}$" in the statement of the principle of optimality. There are some subgames that are never reached with positive probability under an optimal strategy, such as the lower subgame in figure 1 corresponding to the decision $d_1 = 2$ (i.e. move "down"). Thus there are trivial alternative optimal decision rules that do not satisfy the principle of optimality because they involve taking suboptimal decisions on "zero probability subgames." Since these subgames are never reached, such modifications do not jeopardize *ex ante* optimality. However we cannot be sure *ex ante* which subgames will be irrelevant *ex post* unless we carry out the full backward induction process. Dynamic programming results in strategies that are optimal in *every possible subgame,* even those which will never be reached when the strategy is executed. Since backward induction results in a decision rule δ that is optimal for all possible subgames, it is intuitively clear that δ is optimal for the game as a whole, i.e. it a solution to the *ex ante* strategic form of the optimization problem (2).

Theorem 1: *Dynamic programming, i.e. the backward induction process given in equations (4) and (5), results in an optimal decision rule. That is, for each s_1 we have*

$$V_1(s_1) = \max_{\delta\in\mathcal{F}} E\left\{U\left(\{\tilde{s}_t, \tilde{d}_t\}_\delta\right)\right\}. \tag{9}$$

For a formal proof of this result for games against nature (with appropriate care taken to ensure measurability and existence of solutions), see Gihman and Skorohod (1979). However the intuition why backward induction works should be intuitively obvious: backward induction insures that at *every* node of the game tree, the agent selects the action that results in the highest expected payoff for every possible "continuation game". The value functions play the role of *shadow prices,* summarizing the future consequences of taking alternative feasible actions, accounting for all the possible future moves that nature can take.

If in addition to "nature" we extend the game tree by adding another rational expected utility maximizing player, then backward induction can be applied in the same way to solve this alternating move dynamic game. Assume that player 1 moves first, then player 2, then nature, etc. Dynamic programming results in a *pair of strategies* for both players. Nature still plays a "mixed strategy" that could depend on the entire previous history of the game, including all of the previous moves of both players. The backward induction process insures that each player can predict the future choices of their opponent, not only in the succeeding move, but in all future stages of the game. The pair of strategies (δ^1,δ^2) produced by dynamic programming are mutual best responses, as well as being best responses to nature's moves. Thus, these strategies constitute a *Nash equilibrium*. They actually satisfy a stronger condition: they are Nash equilibrium strategies in every possible subgame of the original game, and thus are *subgame-perfect* (Selten, 1975). Subgame-perfect equilibria exclude "implausible equilibria" based on *incredible threats*. A standard example is an incumbent's threat to engage in a price war if a potential entrant enters the market. This threat is incredible if the incumbent would not really find it advantageous to engage in a price war (resulting in losses for both firms) if the entrant called its bluff and entered the market. Thus the set of all Nash equilibria to dynamic multiplayer games is strictly larger than the subset of subgame perfect equilibria, a generalization of the fact that in single agent decision problems, the set of optimal decision rules include ones which take suboptimal decisions on subgames that have zero chance of being reached for a given optimal decision rule. Dynamic programming ensures that the decision maker would never mistakenly reach any such subgame, similar to the way it ensures that a rational player would not be fooled by an incredible threat.

3.4 Dynamic Programming for Stationary, Markovian, Infinite Horizon Problems

The *complexity* of dynamic programming arises from the exponential growth in the number of possible histories as the number of possible values for the state variables, decision variables, and/or number of time periods T increases. For example, in a problem with N possible values for s_t and D possible values for d_t in each time period t, there are $[ND]^T$ possible histories, and thus the required number of calculations to solve a general T period, history-dependent dynamic programming problem is $O([ND]^T)$. Bellman and Dreyfus (1962) referred to this exponential growth in the number of calculations as the *curse of dimensionality*. In the next section I will describe various strategies for dealing with this problem, but an immediate solution is to restrict attention to *time separable Markovian decision problems*. These are problems where the payoff function U is additively separable as in equation (1), and where both the choice sets $\{D_t\}$ and the transition probabilities $\{p_t\}$ only depend on the contemporaneous state variable s_t and not the entire previous history H_{t-1}. We say a conditional distribution p_t satisfies the *Markov property* if it depends on the previous history only via the most recent values, i.e. if $p_t(s_t|H_{t-1}) = p_t(s_t|s_{t-1},d_{t-1})$. In this case backward induction becomes substantially easier. For example in this case the dynamic programming optimizations have to be performed only at each of the N possible values of the state variable at each time t, so only $O(NDT)$ calculations are required to solve a time T period time separable Markovian problem instead of $O([ND]^T)$ calculations when histories matter. This is part of the reason why, even though time nonseparable utilities and non-Markovian forms of uncertainty may be more general, most dynamic programming problems that are solved in practical applications are both time separable and Markovian.

SDPs with random horizons $\tilde{T}$ can be solved by backward induction provided there is some finite time $\overline{T}$ satisfying $Pr\{\tilde{T} \le \overline{T}\} = 1$. In this case, backward induction proceeds from the maximum possible value $\overline{T}$ and the *survival probability* $\rho_t = Pr\{\tilde{T} > t|\tilde{T} \ge t\}$ is used as to capture the probability that the problem will continue for at least one more period. The Bellman equation for the discounted, time-separable utility with uncertain lifetime is

$$\begin{aligned} V_t(s_t) &= \max_{d \in D_t(s_t)} \left[u_t(s_t,d) + \rho_t \beta EV_{t+1}(s_t,d)\right] \\ \delta_t(s_t) &= \underset{d \in D_t(s_t)}{argmax} \left[u_t(s_t,d) + \rho_t \beta EV_{t+1}(s_t,d)\right], \end{aligned} \tag{10}$$

where

$$EV_{t+1}(s,d) = \int_{s'} V_{t+1}(s') p_{t+1}(s'|s,d). \tag{11}$$

In many problems there is no finite upper bound $\overline{T}$ on the horizon. These are called *infinite horizon problems* and they occur frequently in economics. For example, SDPs used to model decisions by firms are typically treated as infinite horizon problems. It is also typical in infinite horizon problems to assume *stationarity*. That is, the utility function $u(s,d)$, the constraint set $D(s)$, the survival probability ρ, and the transition probability $p(s'|s,d)$ do not explicitly depend on time t. In such cases, it is not hard to show the value function and the optimal decision rules are also stationary, and satisfy the following version of Bellman's equation

$$\begin{aligned} V(s) &= \max_{d \in D(s)} \left[u(s,d) + \rho\beta EV(s,d)\right] \\ \delta(s) &= \underset{d \in D(s)}{argmax} \left[u(s,d) + \rho\beta EV(s,d)\right], \end{aligned} \tag{12}$$

where

$$EV(s,d) = \int_{s'} V(s') p(s'|s,d). \tag{13}$$

This is a fully recursive definition of V, and as such there is an issue of existence and uniqueness of a solution. In addition, it is not obvious how to carry out backward induction, since there is no "last" period from which to begin the backward induction process. However under relatively weak assumptions, one can show there is a unique V satisfying the Bellman equation, and the implied decision rule in equation (12) is an optimal decision rule for the problem. Further, this decision rule can be approximated by solving an approximate finite horizon version of the problem by backward induction.

For example, suppose that $u(s,d)$ is a continuous function of (s,d), the state space S is compact, the constraint sets $D(s)$ are compact for each $s \in S$, and the transition probability $p(s'|s,d)$ is weakly continuous in (s,d) (i.e. $EV(s,d) \equiv \int_{s'} W(s')p(s'|s,d)$ is a continuous function of (s,d) for each continuous function $W : S \to R$). Blackwell (1965), Denardo (1967) and others have proved that under these sorts of assumptions, V is the unique fixed point to the *Bellman operator* $\Gamma : B \to B$, where B is the Banach space of continuous functions on S under the supremum norm, and Γ is given by

$$\Gamma(W)(s) = \max_{d \in D(s)} \left[u(s,d) + \rho\beta \int_{s'} W(s')p(s'|s,d)\right]. \tag{14}$$

The existence and uniqueness of V is a consequence of the *contraction mapping theorem,* since Γ can be shown to satisfy the contraction property,

$$\|\Gamma W - \Gamma V\| \leq \alpha \|W - V\|, \tag{15}$$

where $\alpha \in (0,1)$ and $\|W\| = \sup_{s \in S} |W(s)|$. In this case, $\alpha = \rho\beta$, so the Bellman operator will be a contraction mapping if $\rho\beta \in (0,1)$.

The proof of the optimality of the decision rule δ in equation (12) is somewhat more involved. Using the Bellman equation (12), we will show that (see equation (43) in section 4),

$$V(s) = u(s,\delta(s)) + \rho\beta \int_{s'} V(s')p(s'|s,\delta(s)) = E\left\{ \sum_{t=0}^{\infty} [\rho\beta]^t u(s_t,\delta(s_t)) \middle| s_0 = s \right\}, \tag{16}$$

i.e. V is the value function implied by the decision rule δ. Intuitively, the boundedness of the utility function, combined with discounting of future utilities, $\rho\beta \in (0,1)$, implies that if we truncate the infinite horizon problem to a T period problem, the error in doing so would be arbitrarily small when T is sufficiently large. Indeed, this is the key to understanding how to find approximately optimal decision rules to infinite horizon SDPs: *we approximate the infinite horizon decision rule* δ *by solving an approximate finite horizon version of the problem by dynamic programming*. The validity of this approach can be formalized using a well known property of contraction mappings, namely that the *method of successive approximations* starting from any initial guess W converges to the fixed point of Γ, i.e.

$$\lim_{t\to\infty} V_t = \Gamma^t(W) = V \quad \forall W \in B, \tag{17}$$

where $\Gamma^t W$ denotes t successive iterations of the Bellman operator Γ,

$$\begin{aligned} V_0 &= \Gamma^0(W) = W \\ V_1 &= \Gamma^1(W) \\ &\cdots \\ V_t &= \Gamma^t(W) = \Gamma(\Gamma^{t-1}W) = \Gamma(V_{t-1}). \end{aligned} \tag{18}$$

If $W = 0$ (i.e. the zero function in B), then $V_T = \Gamma^T(0)$ is simply the period $t = 1$ value function resulting from the solution of a T period dynamic programming problem. Thus, this result implies that the optimal value function V_T for a T-period approximation to the infinite horizon problem converges to V as $T \to \infty$. Moreover the difference in the two functions satisfies the bound

$$\|V_T - V\| \leq \frac{[\rho\beta]^T \|u\|}{1-\rho\beta}. \tag{19}$$

Let $\delta_T = (\delta_{1,T}, \delta_{2,T}, \ldots, \delta_{T,T})$ be the optimal decision rule to the T period problem. It can be shown that if we follow this decision rule up to period T and then use $\delta_{1,T}$ in every period after T, the resulting decision rule is approximately optimal in the sense that the value function for this infinite horizon problem also satisfies inequality (19), and thus can be made arbitrarily small as T increases.

In many cases in economics the state space S has no natural upper bound. An example might be where s_t denotes an individual's wealth at time t, or the capital stock of the firm. If the unboundedness of the state space results in unbounded payoffs, the contraction mapping argument must be modified since the Banach space structure under the supremum norm no longer applies to unbounded functions. Various alternative approaches have been used to prove existence of optimal decision rules for unbounded problems. One is use an alternative norm (e.g. a *weighted norm*) and demonstrate that the Banach space/contraction mapping argument still applies However there are cases where there are no natural weighted norms, and the contraction mapping property cannot hold since the Bellman equation can be shown to have multiple solutions, see e.g. Rust (2003). The most general conditions under which the existence and uniqueness of the solution V to the Bellman equation and the optimality of the implied stationary decision rule δ has been established is in Bhattacharya and Majumdar (1989). However as I discuss in section 4, considerable care must be taken in solving unbounded problems numerically.

3.5 Application to Dynamic Games, Mechanisms, and Competitive Equilibria

The section briefly discusses how dynamic programming can be used to solve *dynamic games, dynamic mechanisms (contracts),* and *recursive competitive equilibria* of dynamic economies. For a more in depth treatment of these topics, I direct the reader to Ljungqvist and Sargent (2000). Consider first how dynamic programming can be applied to solve dynamic games. To see why dynamic programming applies to dynamic games, consider the decision problem of a particular agent in the game. *In equilibrium,* the strategies of this agent's opponents can be treated as fixed, i.e. as part of the "law of motion" that the agent faces, similar to the transition probability for nature in a single agent decision problem. Just as the optimal decision rule is a best response to the law of motion for nature in a single agent problem, the equilibrium decision rule for each agent must constitute a best response to a composite law of motion resulting from nature's "mixed strategy" and the equilibrium decision rules of the other agents.

However there is additional complexity in the multiagent case due to the fact that in equilibrium, the strategies of the rational players must be *mutual best responses*. In a static game, the problem of finding Nash equilibrium strategies can be recast mathematically as a *fixed point problem*. In a stationary, infinite horizon context, an equilibrium to a dynamic multiplayer game can be viewed as a *nested fixed point problem*. That is, for any given profile of strategies for opponents, we need to solve an *inner dynamic programming/contraction mapping fixed point problem* to determine any particular agent's best response to the other strategies. The *outer fixed point problem* iteratively searches for a profile of strategies that are mutual best responses for all players in the game.

A second complication and difference between dynamic multiplayer games and single agent decision problems, is the issue of *multiple equilibria*. Although multiple equilibria arise even in single agent SDPs, i.e. there can be multiple optimal decision rules, in single agent problems the agent receives the same expected payoff from any optimal decision rule, so multiplicity is not an issue of consequence. However it is a significant issue in multiagent SDPs because in this setting different equilibria can result in *different* payoffs for the agents in the game. There are very complicated issues about *beliefs* players have about what strategies their opponents will adopt, as well as how much history should be used when agents formulate their strategies. At the most extreme, the literature on *repeated games* shows that when there is no restrictions placed on these histories, there is a huge multiplicity of equilibria – often there are a continuum of equilibria in such games. The literature on *Folk Theorems* has shown that if players are sufficiently patient, the set of possible equilibria is so large that essentially "anything" can happen.[11]

To reduce the set of equilibria, the game theory literature as investigated various *equilibrium refinements* and restrictions on "payoff-relevant histories." A prominent example is the class of *Markov-perfect equilibria* (Maskin and Tirole, 2001), where the payoff relevant history is assumed to be the current state s_t which evolves according to a Markov transition probability. This probability depends on joint set of actions of all the players (who are assumed to move simultaneously at each time t), i.e. $p(s_{t+1}|s_t,d_t)$, where $d_t = (d_{t,1},\dots,d_{t,n})$ is the vector of decisions taken by the n players in the game at time t. Further, the set of feasible actions for each player is also assumed to depend only on s_t, so $D_i(s_t)$ is the state-dependent constraint set for player i in state s_t. A profile of strategies $\delta = (\delta_1,\dots,\delta_n)$ constitutes a (pure strategy) *Markov-perfect Nash equilibrium* if for each $i \in \{1,\dots,n\}$ the following set of Bellman equations hold

$$V_i(s) \quad = \quad \max_{d_i \in D_i(s)} \left[u_i(s,d_i,\delta_{-i}(s)) + \beta \int_{s'} V_i(s')p(s'|s,d_i,\delta_{-i}(s)) \right]$$

[11] Specifically any "undominated" profile of payoffs to the players can be an equilibrium in the repeated game. A payoff is undominated if it exceeds the player's *maximin payoff,* which is the payoff each player could unilaterally assure themselves even in the worst case when the player's opponents adopt strategies that result in the worst possible payoff to the player in question.

$$\delta_i(s) \quad = \quad \underset{d_i \in D_i(s)}{argmax} \left[u_i(s,d_i,\delta_{-i}(s)) + \beta \int_{s'} V_i(s')p(s'|s,d_i,\delta_{-i}(s)) \right], \qquad (20)$$

where $\delta_{-i} = (\delta_1,\dots,\delta_{i-1},\delta_{i+1},\dots,\delta_n)$, i.e. the profile of strategies for all of the opponents to player i. Thus, we can see that for any fixed δ_{-i}, the Bellman equation (20) is the same as an ordinary single agent infinite horizon Markovian decision problem that we discussed in the previous section. This is the "inner" contraction fixed point problem of the nested fixed point problem. The "outer" fixed point problem is to find a set of decision rules $(\delta_1,\dots,\delta_n)$ that are mutual best responses. Algorithms for computing Markov-perfect Nash equilibria typically iterate between the inner and outer fixed point problems, and use dynamic programming to repeatedly solve the inner fixed point problem, i.e. the Bellman equations (20), until a set of mutual best responses has been found. See Pakes and McGuire (1994) for examples of algorithms that compute Markov-perfect equilibria.

Nested fixed point algorithms, using dynamic programming to solve the inner Bellman equation fixed point problem, are also used to compute *recursive competitive equilibria* of dynamic economies. These are very similar to dynamic games, except that these models of dynamic economies that typically assume there are a continuum of *atomistic* agents each of who has negligible influence on prices, equilibrium quantities, etc. The problem of interest is to compute a set of state-contingent equilibrium prices and quantities for these economies (and with a continuum of agents, these equilibrium objects are typically *functions* or *distributions*) that satisfy 1) market clearing, and 2) *rational expectations*. The latter condition states that the distribution of prices that agents expect to hold in the economy (and which constitutes the beliefs upon which agents take actions), is the same as the prices that actually hold in equilibrium. Many recursive competitive equilibria are based on a *representative agent paradigm* where the utility, profit or payoff functions of all agents are the same, and agents are differentiated only by the fact that each as different values for their state (e.g. savings, capital stocks, etc.). The Lucas (1978) asset pricing model is an example of the representative agent framework. However there are heterogeneous agent versions of recursive competitive equilibria, where different agents have different utility or profit functions. For example Rust (1983) uses dynamic programming methods to solve an equilibrium model of durable goods pricing where agents are heterogeneous with respect to their relative preferences for new versus used durable goods. The basic structure of the nested fixed point algorithm in such problems is as follows: the "outer" fixed point algorithm searches for a distribution of prices (or whatever other objects are part of the definition of the recursive competitive equilibrium, such as distributions of quantities, etc.) that satisfies the rational expectations and market clearing conditions, whereas for each trial value of these objects, dynamic programming is used to solve the inner fixed point problem, which results in solutions for the individual agent Bellman equations for the corresponding optimal decision rules. The individual decisions are then aggregated (or integrated, in the case of continuum agent economies) and this information is fed back to the outer fixed point algorithm to be used to check whether the current iteration is an approximate equilibrium or not.

Recursive competitive equilibria can become quite complicated when there are a continuum of goods and *macro shocks* that have persistent effects on the economy. In many cases, the macro shock alone is not a "sufficient statistic" for the future evolution of the economy: one also needs to account for the entire *distribution of quantities* and how this distribution will evolve over time. This distribution then becomes an infinite dimensional state variable that must be carried around to solve individual agent dynamic programming problems. For example, consider automobiles and let the macro shock be the price of gasoline. Sustained periods of high gas prices lead consumers to scrap older vehicles, and this truncates the distribution of vintages in future years. Thus it is necessary to account for the entire distribution of vintages to predict future prices of used cars. Krusell and Smith (1998) consider a model of investment by firms in the presence of macro shocks that affect firm demand. In this case, aggregate demand for capital (which determines the interest rate) depends not only on the current macro shock, but on the entire distribution of capital in the economy. Fortunately, there are computational methods (to be discussed in section 3) that makes it possible to approximate the solution to these problems by *parameterizing* these infinite-dimensional state variables using the the corresponding vector of parameters as representing the relevant "state".[12]

In recent years recursive dynamic programming methods have been extended and adapted to provide constructive means for solving *dynamic mechanism design problems* and to characterize the subset of subgame perfect (or *sequential*) equilibria of repeated games. Due to space limitations I cannot describe these extensions in much detail, except to state that both developments are applicable to problems with *asymmetric information* and make ingenious use of the concept of a *promised utility* as a "state variable" as an efficient, recursive means for characterizing and solving what would otherwise be extremely complicated history-dependent strategies. Dynamic mechanism design problems can be viewed as a type of Stackelberg game, where the Stackelberg leader (SL) chooses a *mechanism* (i.e. the "rules of the game") in order to maximize some objective function. The difference is that the SL has incomplete information about the Stackelberg followers (SF). Myerson (1981) has shown that optimal mechanisms can be characterized as the solution to a linear programming problem, where *incentive constraints* are used to induce truthful revelation of private information by SF, a consequence of the *revelation principle*.[13] In some problems (e.g. optimal auction design) the optimal mechanism is fundamentally *static*, but in other cases the the SL can achieve better outcomes by constructing *dynamic mechanisms* because history-dependent mechanisms can generally enable the SL to obtain more control over the behavior of SFs. However a "direct" formulation of the dynamic mechanism design problem that directly optimizes over the space of all history-dependent mechanisms is subject to the same curse of dimensionality problem as we observed in history-dependent SDPs in (2): the space of possible strategies (mechanisms) is "too large" for a direct approach to be tractable. However Spear and Srivastava (1987) showed that an optimal dynamic mechanism can be found within the subclass of *recursive mechanisms*. Building on similar ideas that originated Abreu, Pearce and Stachetti (1986), they showed that promised utility (the expected discounted utility that SFs expect to receive at any point in the "continuation game") can be treated as both a state and control variable under (indirect) control of the SL. The SL chooses the promised utilities subject to incentive constraints that induce the SF to truthfully reveal their private information at every stage of the dynamic programming problem, and also "promise keeping" constraints that guarantee that the current promised utility is the expected value of promised utilities in the "continuation game". This approach has been successfully applied to solve a large number of dynamic contracting and incentive problems. Abreu, Pearce and Stacchetti (1990) applied recursive methods with promised utilities in order to characterize the *equilibrium payoff set,* i.e. the set of all sequential equilibrium payoffs in repeated games where agents have imperfect information about each other's actions. They introduced the concept of *self-generation* and showed that the equilibrium payoff set is the largest self-generating set, a type of "set valued" fixed point condition. Their methods have been applied to numerically calculate equilibrium payoff sets by Judd, Yeltekin and Conklin (2003). See Ljungqvist and Sargent (2000) for further applications and illustrations

[12]In particular, in the Krusell and Smith case, they found that the mean capital stock was an approximate "sufficient statistic" for the entire distribution, so it was only necessary to develop a law of motion that predicts how mean capital co-evolves with macro shocks rather than having to model how the entire distribution of capital in the economy evolves over time.

[13]The revelation principle states that the Stackelberg leader can find the optimal (information-constrained) mechanism within the class of *direct revelation games,* with incentive constraints that induce SFs to truthfully reveal their private information.

of these methods.

3.6 Time Consistency and the Limits to Dynamic Programming

A key property of solutions (e.g. decision rules, equilibria, etc.) produced by dynamic programming is that they are *time consistent,* that is, by construction, the agent has no *ex post* incentive to deviate from any strategy that was calculated by backward induction *ex ante* (i.e. before any decisions are actually made). This is essentially a direct implication of the principle of optimality, or the concept of subgame perfection in games. However validity of backward induction, particularly that it yields an optimal solution to the strategic formulation of the SDP (Theorem 1), is closely connected with the *linearity properties* of expectations and conditional expectations, and thus, its validity is closely connected to the assumption that decision makers are expected utility maximizers. In particular, the proof of Theorem 1 makes use of the *Law of Iterated Expectations* to ensure that the dynamic programming solution is indeed a solution to the *ex ante* strategic formulation of the SDP or game. However as Machina (1989) and others have noted, the linearity property of expected utility appears to be a key property that is violated in numerous experimental studies that have found that human decision making under uncertainty is inconsistent with expected utility maximization. Instead the literature on *non-expected utility* has suggested alternative formulations of preferences that are *nonlinear functions* of the probabilities representing the uncertainty in a decision problem. However this nonlinearity can invalidate the use of dynamic programming, which can result in *ex ante suboptimal* decision rules for the strategic formulation of the SDP (e.g. equation (2)).

For example, suppose an individual is a *median utility maximizer* instead of an expected utility maximizer. That is, they choose a strategy that results in the highest *median utility.* Dynamic programming proceeds just as it does for the expected utility case, expect at each stage we choose the action that maximizes the *conditional median* of future utility. Unfortunately, it is easy to construct examples where dynamic programming results in a *suboptimal decision rule* when this decision rule is evaluated in terms of the *ex ante* strategic formulation of the problem.[14] The failure of dynamic programming is a consequence of the nonlinearity of the median as an "aggregator function." In particular, the median of the sum of two random variables is not equal to the sum of the individual medians, and there is no "law of the iterated median." In a more complicated example, Nielsen and Jaffray (2004) have shown that "naive" applications of backward induction fail in a more egregious fashion for individuals who have *rank dependent utility* (RDU). RDU preferences were introduced as a potential solution to the *Allais paradox,* one of the most famous experimental violations of expected utility theory. The authors show that decision rules produced by backward induction are not only suboptimal, they are *dominated* by other decision rules in

[14]Consider a two stage game where the agent moves first, then nature moves. At the first stage the agent can move down (D) to get a sure payoff of 5.1 and the game ends. If the agent chooses up (U), nature moves next and randomly chooses U or D with probability 1/2. If nature chooses D, the agent gets a payoff of 10 and the game ends. If nature chooses U, the agent faces a final U or D decision. If the agent chooses U in this final subgame, nature makes the last move and selects a random payoff for the agent equal to -3 with probability .4, 4 with probability .2 or 10 with probability .4. If the agent chooses D, nature's final move results in payoffs to the agent of -10 with probability .4, 5 with probability .2 and 10 with probability .4. The (conditional) median utility to the agent from choosing D in this final subgame is 5 (since 5 is the median outcome from nature's move), whereas choosing D results in a median utility of 4, so the value of the subgame reached when the agent's initial decision is U is 5. Since nature's first move is U or D with probability 1/2, the median utility to the agent of choosing U in the first move is 5, which is lower than the median utility of choosing D (5.1). Thus, the dynamic programming solution is for the agent to choose D in the first stage for a payoff of 5.1 and the game immediately ends. However from an *ex ante* perspective, the strategy (U,D) (moving U in the first stage and D in the second) results in a random payoff of -10 with probability .2, 5 with probability .1 and 10 with probability .7. The median of this random payoff is 10, so it follows that dynamic programming has resulted in a suboptimal decision rule since it results in a median payoff of only 5.1 whereas (U,D) results in a median payoff of 10.

terms of first order stochastic dominance.

In fact, Hammond (1988) proved that under certain assumptions, particularly the assumption of *consequentialism,* expected utility is the *only* form of preferences that result in time consistent behavior, and thus, for which dynamic programming can be guaranteed to produce strategies that are *ex ante* optimal in terms of the strategic formulation of the decision problem. The principle of consequentialism states that within any subgame, a decision maker considers only payoffs that can be realized within that subgame: payoffs from *unrealized* subgames, (i.e. outcomes that cannot be attained starting from the current subgame) are *irrelevant* in the sense that they have no effect on the agent's payoffs and their choices in their current subgame.[15]

Machina's (1989) response is that "it is *inappropriate* to impose the property of consequentialism on non-expected utility maximizers." Machina argues that "consequentialism is essentially a dynamic version of the very separability that non-expected utility maximizers reject, and that assuming it in this context is much like assuming, say, that agents with intertemporally non-separable preferences would neglect their consumption histories when making subsequent decisions." (p. 1642-1643). Machina suggests ways to properly formulate a non-expected utility maximizer's evaluation of subgames so their behavior is time-consistent. Thus backward induction using an appropriately modified notion of "conditional preferences" will result in a decision rule that is *ex ante* optimal in terms of the strategic formulation of the problem. However these conditional preferences will not obey the principle of consequentialism, i.e. the individual's conditional preference ordering over any subgame may depend on "unrealized outcomes" from unrealized subgames. In addition, is not always obvious how to define "conditional preferences" so that dynamic programming can be used to find an *ex ante* optimal decision rule. Even when conditional preferences can be defined, the resulting form of backward induction may not bear much resemblance to "traditional" dynamic programming for expected utility maximizers. For example Nielsen and Jaffray's (2004) dynamic programming algorithm for RDU-maximizers "works by backward induction in the tree, but, due to the fact that RDU preferences are not dynamically consistent, departs significantly from the standard dynamic programming algorithm; in particular, when a decision node is reached: instead of selecting one substrategy, a set of 'acceptable' substrategies is selected, and the final choice among them is made at the next step." (p. 6). The only class of non-time-separable, non-expected utility preferences for which I am aware that dynamic programming applies in a "straightforward" fashion is the class of *recursive preferences*.

Recursive utility, initially introduced by Koopmans (1960) in a deterministic setting, is a convenient way to produce intertemporally non-additive utility functions. In an infinite horizon context, recursive utility functions are defined, as their name suggests, *recursively* via the solution to the equation

$$U(s_1,d_1,\dots) = W(s_1,d_1,U(s_2,d_2,\dots,)) \tag{21}$$

where W is an *aggregator function*. Note that time-separable discounted utility is a special case when $W(s_1,d_1,U) = u(s_1,d_1)+\rho\beta U$. The aggregator function W is typically assumed to satisfy a contraction mapping condition in its second argument, i.e. for some $\alpha \in (0,1)$, $|W(s,d,U)-W(s,d,V)| \leq \alpha|U-V|$, which is sufficient (via the contraction mapping theorem) to guarantee the existence and uniqueness of an infinite horizon utility function U that solves Koopman's equation (21). Recursive preferences can be extended to SDP problems involving uncertainty by introducing a second aggregator A, known as a

[15]Or as Machina (1989) puts it, "In other words, the only determinants of how decisions should be made in the continuation of a decision tree are the *original* preference ordering over probability distributions and the attributes of the *continuation* of the tree." (p. 1641). See also Epstein and Le Breton (1993) who showed that only class of belief updating rules that are dynamically consistent are those that satisfy *Bayes rule*. This suggests that time consistent behavior is uniquely associated with expected utility maximization and Bayesian updating of beliefs, and this is precisely the framework used in "standard" game theoretic models.

stochastic aggregator or *certainty equivalent*. A prominent special case is where $A = E$, the conditional expectation operator corresponding to a stationary transition probability $p(s'|s,d)$. However, following Epstein and Zin (1989) and Skiadas (1997) other types of stochastic aggregators can be considered some of which are "nonlinear in probabilities." One example is the min function, which can represent a "worst case" approach to the evaluation of uncertain outcomes. Given any stationary, feasible decision rule δ (i.e. $\delta(s) \in D(s)$, $\forall s \in S$), define the agent's *recursive utility function* $V_\delta : S \to R$ by

$$V_\delta(s) = W(s, \delta(s), AV_\delta(s, \delta(s))). \tag{22}$$

Note that if W is nonlinear in its second argument, even if the stochastic aggregator A is the conditional expectation operator, this recursively defined utility function is generally not consistent with expected utility maximization. Skiadis (1997) provides general conditions sufficient to guarantee the existence of V_δ. A sufficient condition for existence and uniqueness is the case $A = E$, where E is the conditional expectation operator induced by $p(s'|s,d)$, and where W satisfies the contraction mapping property in its second argument. For more general stochastic aggregators A it may not be possible to establish the existence and uniqueness of V_δ using the contraction mapping theorem, but for purposes of the discussion here, suppose that the mapping Γ_δ defined by

$$\Gamma_\delta(V)(s) = W(s, \delta(s), AV(s, \delta(s))) \tag{23}$$

is a contraction mapping, and in particular that V is a continuous and bounded function of s and the state space S and constraint sets $D(s)$ are compact. Then for this class of preferences, the natural generalization of the Bellman equation is

$$V(s) = \max_{d \in D(s)} W(s, d, AV(s,d)), \tag{24}$$

and it is not hard to see that V is also a fixed point of a contraction mapping $\Gamma : B \to B$ (where B is the Banach space of continuous, bounded functions from S to R), defined by

$$\Gamma(V)(s) = \max_{d \in D(s)} W(s, d, AV(s,d)). \tag{25}$$

Let δ be the stationary decision rule implied by V, i.e.

$$\delta(s) = \underset{d \in D(s)}{argmax}\, W(s, d, AV(s,d)). \tag{26}$$

It is not hard to show that δ is the optimal stationary decision rule for this problem, i.e. we can show that for any other stationary decision rule δ' we have $V(s) \geq V_{\delta'}(s)$, for all $s \in S$.[16] Thus, the upshot of this discussion is that there do exist non-trivial classes of non-time separable, non-expected utility preferences for which dynamic programming "works" and results in an *ex ante* optimal decision rule.[17]

[16]The argument requires that A is a *monotone operator* i.e. if $V \geq U$ then $AV \geq AU$, which is true in the case where $A = E$, the conditional expectation operator. If W is monotone in its second argument, this implies that Γ is a monotone operator. The argument for the optimality of δ then follows from the easily verified inequality $V_{\delta'}(s) \leq \Gamma(V_{\delta'})(s)$. Iterating on this, using the monotonicity of Γ we have $V_{\delta'} \leq \Gamma(V_{\delta'}) \leq \Gamma^2(V_{\delta'}) \leq \cdots \leq \Gamma^t(V_{\delta'})$ Since $\lim_{t\to\infty} \Gamma^t(V_{\delta'}) = V = \Gamma(V)$ by the contraction property for Γ it follows that for any δ' we have $V_{\delta'} \leq V = V_\delta$, i.e. δ is an optimal stationary decision rule.

[17]Another class of non-expected utility preferences for which dynamic programming "works" is presented in Epstein and Schneider (2003). They axiomatize a recursive model of utility in a setting where there is 'ambiguity', i.e. the individual is not sure about the probability distribution governing states of nature. Under their assumptions, the decision maker's behavior is both dynamically consistent, and satisfies the property of consequentialism. Their method for constructing sets of priors (representing possible beliefs over unknown parameters in the probability distribution for "nature") implies a version of the law of iterated expectations. As a result, they are able to show that dynamic programming produces *ex ante* optimal decision rules.

In multiplayer games, and dynamic equilibrium where there are multiple rational players, there is a different type of time inconsistency that has received considerable attention in the literature. Kydland and Prescott (1977) defined a notion of time-consistency in what effectively is a two player dynamic game (example, one player is the government, and the other is firms in the economy). Their definition of time consistency is essentially the same one as noted above, i.e. a strategy is time consistent if the player has no *ex post* incentive to deviate from it. Kydland and Prescott observed that in dynamic games, the *ex ante* optimal decision rule is often not time consistent (or in game theoretic terminology, it is not subgame perfect). Their finding is common in *Stackelberg games* where one of the players is the Stackelberg leader (SL) and the others are Stackelberg followers (SF). It if often *ex ante* optimal for the SL to commit to a fixed, time invariant decision rule that induces desired behaviors in the SFs. For example, if the SL is the government interested in setting a tax rate to maximize its tax revenues, the *ex ante* optimal tax rate would be a fixed, time-invariant, and moderate tax on firms' capital stock. This moderate tax provides enough incentive for firms to accumulate capital (or individuals to save, if the tax is on individuals), and this policy maximizes the government's tax revenue *within the class of all time invariant policies*. However the time inconsistency arises from the fact that after "announcing" this moderate tax rate, the government always has a temptation to unexpectedly depart from its *ex ante* optimal plan, "fooling" firms and consumers by suddenly seizing all of their capital/savings, and thereby reneging on its "commitment" to maintain moderate tax rates for the indefinite future. However while there is this temptation to "cheat", if the government ever broke its commitment this way, consumers and firms having been once "burned" would be unwilling to accumulate significant taxable capital or savings in the future, fearing future confiscations. However the government's immediate temptation to cheat might exceed its expected future losses, and hence a deviation from a supposedly time-invariant optimal plan can occur. Phelan (2006) shows that the unique subgame perfect Markovian equilibrium to this game involves periodic confiscations, followed by long periods of moderate tax rates that facilitate a gradual restoration of trust by individuals and firms. There are lower savings by firms and consumers, and lower discounted tax revenues to the government in this subgame-perfect equilibrium, so that the government would in fact be better off if it could credibly commit to a moderate, time invariant capital/wealth tax rate.

Beginning with the work of Strotz (1956) a literature on "hyperbolic discounting" has evolved that intentionally specifies forms of time inconsistent preferences. The interest in such preferences arises from experimental anomalies such as "preference reversals" where subjects are offered choices between a small early reward and a larger later reward (Kirby and Herrnstein, 1995). These studies find that many subjects prefer the earlier reward if the payoff is immediate, but the later reward if both rewards are provided only after some delay. This behavior is hard to reconcile if we assume individuals have time additive discounted utility functions. To account for this and other types of time inconsistency that we seem to see in ourselves or others (such as the continual temptation to defect from a diet, or to fail to save even though committing to a long run saving plan seems *ex ante* optimal, etc.), this literature *assumes* that individuals have time inconsistent preferences. For example, in the hyperbolic discounting literature, preferences are intentionally specified to have a non-recursive form: at time $t = 1$ the preferences are assumed to be $u(s_1,d_1)+\gamma\sum_{t=2}^{\infty}\beta^{t-1}u(s_t,d_t)$, where $\gamma\in(0,1)$ represents "addition impatience to consume" above and beyond that reflected in the discount factor β. The preferences governing behavior at any time $t > 1$ are given by $u(s_t,d_t)+\gamma\sum_{j=t+1}^{\infty}\beta^{j-t}u(s_j,d_j)$. As noted by Gul and Pesendorfer (2005) unless $\gamma = 1$ these preferences are non-recursive since "at time $t > 1$, the individual's period t preferences differ from his *conditional* preferences — the preferences over continuation plans implied by his first period preferences and choices prior to period t. The preference change is taken as a primitive to derive the individual's desire for commitment." (p. 120). This type of "intentionally generated" time inconsistency is different from the type of time inconsistency that we noted can exist for certain types of non-expected utility maximizers. Instead the time inconsistency arises because on closer scrutiny, what is purported to be a "single agent" decision problem is actually a *dynamic game among one's current and future selves*. In this literature, an assumed pattern of time-inconsistent changes in preferences create what is effectively a multiplayer game. As Gul and Pesendorfer's work indicates, it may not be necessary to intentionally introduce this extra complexity in order to provide an explanation for experimental anomalies. They introduce a class of recursive and separable (and thus time-consistent) class of preferences called *dynamic self-control preferences* which can explain many of the experimental anomalies that motivated researchers to consider hyperbolic discounting and other types of time-inconsistent preferences. They note that a drawback of the latter approach is the "multiplicity of subgame perfect Nash equilibria implies that a given decision problem may not have a unique payoff associated with it." and that "this multiplicity is more difficult to understand within the context of a single person decision problem, even if the person in question is dynamically inconsistent." (p. 145).

To summarize this discussion, while we have seen problems where dynamic programming will not lead to a "correct" solution of a SDP or dynamic game because the intended solution to the SDP or game is fundamentally time-inconsistent, there is a broad class of preferences, *recursive preferences,* for which dynamic programming does result in an *ex ante* optimal decision rule.

4 Numerical Dynamic Programming and The Curse of Dimensionality

The previous section showed that dynamic programming is a powerful tool that has enabled us to formulate and solve a wide range of economic models involving sequential decision making under uncertainty — at least "in theory". Unfortunately, the cases where dynamic programming results in *analytical, closed-form solutions* are rare and often rather fragile in the sense that small changes in the formulation of a problem can destroy the ability to obtain an analytic solution. However even though most problems do not have analytical solutions, the theorems in the previous section guarantee the *existence* of solutions, and these solutions can be calculated (or approximated) by numerical methods. In the last several decades, faster computers and better numerical methods have made dynamic programming a tool of substantial practical value by significantly expanding the range of problems that can be solved. In particular, it has lead to the development of a large and rapidly growing literature on econometric estimation and testing of "dynamic structural models" that I will discuss in the next section.

However there are still many difficult challenges that prevent us from formulating and solving models that are as detailed and realistic as we might like, a problem that is especially acute in empirical applications. The principal challenge is what Bellman and Dreyfus (1962) called *the curse of dimensionality*. We have already illustrated this problem in section 3.4: for history dependent SDPs with a finite horizon T and a finite number of states N and actions D, dynamic programming requires $O([ND]^T)$ operations to find a solution. Thus it appears that the time required to compute a solution via dynamic programming increases *exponentially fast* with the number of possible decisions or states in a dynamic programming problem.

Fortunately computer power (e.g. operations per second) has also been growing exponentially fast, a consequence of *Moore's Law* and other developments in information technology, such as improved communications and massive parallel processing.[18] In addition to faster "hardware", research on numerical

[18]Bellman and Dreyfus (1962) carried out calculations on RAND's "Johnniac" computer (named in honor of Jon von Neumann whose work contributed to the development the first electronic computers) and reported that this machine could do 12,500 additions per second. Nowadays a typical laptop computer can do over a billion operations per second and we now have supercomputers that are approaching a *thousand trillion operations per second* (a level known as a "petaflop").

methods has resulted in significantly better "software" that has had a huge impact on the spread of numerical dynamic programming and on the range of problems we can solve. In particular, algorithms have been developed that succeed in "breaking" the curse of dimensionality, enabling us to solve in polynomial time classes of problems that were previously believed to be solvable only in exponential time. The key to breaking the curse of dimensionality is the ability to recognize and exploit *special structure* in an SDP problem. We have already illustrated an example of this in section 3.4: if the SDP is Markovian and utility is time separable, a finite horizon, finite state SDP can be solved by dynamic programming in only $O(NDT)$ operations, compared to the $O([ND]^T)$ operations that are required in the general history-dependent case. There is only enough space here to discuss several of the most commonly used and most effective numerical methods for solving different types of SDPs by dynamic programming. I refer the reader to Puterman (1994), Rust (1996), and Judd (1998) for more in depth surveys on the literature on numerical dynamic programming.

Naturally, the numerical method that is appropriate or "best" depends on the type of problem being solved. Different methods are applicable depending on whether the problem has a) finite vs infinite horizon, b) finite vs continuous-valued state and decision variables, and c) single vs multiple players. In finite horizon problems, backward induction is the essentially the only approach, although as we will see there are many different choices about how to most implement it most efficiently — especially in discrete problems where the number of possible values for the state variables are huge (e.g. chess) or in problems with continuous state variables. In the latter case, it is clearly not possible to carry out backward induction for every possible history (or value of the state variable at stage t if the problem is Markovian and time separable), since there are infinitely many (indeed a continuum) of them. In these cases, it is necessary to *interpolate* the value function, whose values are only explicitly computed at a finite number of points in the state space. I use the term *grid* to refer to the finite number of points in the state space where the backward induction calculations are actually performed. Grids might be *lattices* (i.e. regularly spaced sets of points formed as cartesian products of unidimensional grids for each of the continuous state variables), or they may be *quasi-random grids* formed by randomly sampling the state space from some probability distribution, or by generating deterministic sequences of points such as *low discrepancy sequences*. The reason why one might choose a random or low discrepancy grids instead of regularly spaced lattice is to break the curse of dimensionality, as I discuss shortly. Also, in many cases it is advantageous to refine the grid over the course of the backward induction process, starting out with an initial "coarse" grid with relatively few points and subsequently increasing the number of points in the grid as the backward induction progresses. I will have more to say about such *multigrid* and *adaptive grid* methods when I discuss solution of infinite horizon problems below.

Once a particular grid is chosen, the backward induction process is carried out in the normal way it would be done in a finite state problem. Assuming the problem is Markovian and the utility it time separable and there are n grid points $\{s_1,\dots,s_n\}$, this involves the following calculation at each grid point s_i, $i=1,\dots,n$

$$V_t(s_i) = \max_{d\in D_t(s_i)} \left[u_t(s_i,d) + \rho\beta \hat{E}V_{t+1}(s_i,d)\right], \tag{27}$$

where $\hat{E}V_{t+1}(s_i,d)$ is a numerical estimate of the conditional expectation of next period's value function. I will be more specific about what numerical integration methods methods are appropriate below, but at this point is suffices to note that all of them are simple weighted sums of values of the value function at $t+1$, $V_{t+1}(s)$. We can now see that even if the actual backward induction calculations are carried out only at the n grid points $\{s_1,\dots,s_n\}$, we will still have to do numerical integration to compute $\hat{E}V_{t+1}(s_i,d)$ and the latter calculation may require values of $V_{t+1}(s)$ at points s off the grid, i.e. at points $s \notin \{s_1,\dots,s_n\}$.

This is why some form or interpolation (or in some cases *extrapolation*) is typically required. Almost all methods of interpolation can be represented as weighted sums of the value function at its known values $\{V_{t+1}(s_1),\dots,V_{t+1}(s_n)\}$ at the n grid points, which were calculated by backward induction at the previous stage. Thus, we have

$$\hat{V}_{t+1}(s) = \sum_{j=1}^{n} w_i(s) V_{t+1}(s_i), \tag{28}$$

where $w_i(s)$ is a weight assigned to the i^{th} grid point that depends on the point s in question. These weights are typically positive and sum to 1. For example in *multilinear interpolation* or *simplicial interpolation* the $w_i(s)$ weights are those that allow s to be represented as a convex combination of the vertices of the smallest lattice hypercube containing s. Thus, the weights $w_i(s)$ will be zero for all i except the immediate neighbors of the point s. In other cases, such as *kernel density* and *local linear regression* the weights $w_i(s)$ are generally nonzero for all i, but the weights will be highest for the grid points $\{s_1 \dots, s_n\}$ which are the *nearest neighbors* of s. An alternative approach can be described as *curve fitting*. Instead of attempting to interpolate the calculated values of the value function at the grid points, this approach treats these values as a *data set* and estimates parameters θ of a flexible functional form approximation to $V_{t+1}(s)$ by *nonlinear regression*. Using the estimated $\hat{\theta}_{t+1}$ from this nonlinear regression, we can "predict" the value of $V_{t+1}(s)$ at any $s \in S$

$$\hat{V}_{t+1}(s) = f(s, \hat{\theta}_{t+1}). \tag{29}$$

A frequently used example of this approach is to approximate $V_{t+1}(s)$ as a linear combination of K "basis functions" $\{b_1(s),\dots,b_K(s)\}$. This implies that $f(s,\theta)$ takes the form of a *linear regression* function

$$f(s,\theta) = \sum_{k=1}^{K} \theta_k b_k(s), \tag{30}$$

and $\hat{\theta}_{t+1}$ can be estimated by *ordinary least squares*. *Neural networks* are an example where f depends on θ in a nonlinear fashion. Partition θ into subvectors $\theta = (\gamma,\lambda,\alpha)$, where γ and λ are vectors in R^J, and $\alpha = (\alpha_1,\dots,\alpha_J)$, where each α_j has the same dimension as the state vector s. Then the neural network f is given by

$$f(s,\theta) = f(s,\gamma,\lambda,\alpha) = \sum_{j=1}^{J} \gamma_j \phi(\lambda_j + \langle s, \alpha_i \rangle) \tag{31}$$

where $\langle s, \alpha_j \rangle$ is the inner product between s and the conformable vector α_j, and ϕ is a "squashing function" such as the logistic function $\phi(x) = \exp\{x\}/(1+\exp\{x\})$. Neural networks are known to be "universal approximators" and require relatively few parameters to provide good approximations to nonlinear functions of many variables. For further details on how neural networks are applied, see the book by Bertsekas and Tsitsiklis (1996) on *neuro-dynamic programming*.

All of these methods require extreme care for problems with *unbounded state spaces*. By definition, any finite grid can only cover a small subset of the state space in this case, and thus any of the methods discussed above would require *extrapolation* of the value function to predict its values in regions where there are no grid points, and thus "data" on what its proper values should be. Mistakes that lead to incorrect extrapolations in these regions can not only lead to errors in the regions where there are no grid points, but the errors can "unravel" and also lead to considerable errors in approximating the value function in regions where we do have grid points. Attempts to "compactify" an unbounded problem by arbitrarily truncating the state space may also lead to inaccurate solutions, since the truncation is itself an implicit form of extrapolation (e.g. some assumption needs to be made what to do when state variables approach the "boundary" of the state space: do we assume a "reflecting boundary", an "absorbing boundary", etc.?). For example in life cycle optimization problems, there is no natural upper bound on wealth, even if it is true that there is only a finite amount of wealth in the entire economy. We can always ask the question, if a person had wealth near the "upper bound", what would happen to next period wealth if he invested some of it? Here we can see that if we extrapolate the value function by assuming that the value function is bounded in wealth, this means that by definition there is no incremental return to saving as we approach the upper bound. This leads to lower saving, and this generally leads to errors in the calculated value function and decision rule far below the assumed upper bound. There is no good general solution to this problem except to solve the problem on a much bigger (bounded) state space than one would expect to encounter in practice, in the hope that extrapolation induced errors in approximating the value function die out the further one is from the boundary. This property should hold for problems where the probability that the next period state will hit or exceed the "truncation boundary" gets small the farther the current state is from this boundary.

When a method for interpolating/extrapolating the value function has been determined, a second choice must be made about the appropriate method for *numerical integration* in order to approximate the conditional expectation of the value function $EV_{t+1}(s,d)$ given by

$$EV_{t+1}(s,d) = \int_{s'} V_{t+1}(s')p_{t+1}(s'|s,d). \tag{32}$$

There are two main choices here: 1) deterministic quadrature rules, or 2) (quasi) Monte carlo methods. Both methods can be written as weighted averages of form

$$\hat{E}V_{t+1}(s,d) = \sum_{i=1}^{N} w_i(s,d)V_{t+1}(a_i), \tag{33}$$

where $\{w_i(s,d)\}$ are *weights,* and $\{a_i\}$ are *quadrature abscissae*. Deterministic quadrature methods are highly accurate (for example an N-point Gaussian quadrature rule is constructed to exactly integrate all polynomials of degree $2N-1$ or less), but become unwieldy in multivariate integration problems when *product rules* (tensor products of unidimensional quadrature) are used. *Any* sort of deterministic quadrature methods can be shown to be subject to the curse of dimensionality in terms of worst case computational complexity (see Traub and Werschulz 1998). For example if $N = O(1/\varepsilon)$ quadrature points are necessary to approximate a univariate integral within ε, then in a d-dimensional integration problem $N^d = O(1/\varepsilon^d)$ quadrature points would be necessary to approximate the integral with an error of ε, which implies that computational effort to find an ε-approximation increases exponentially fast in the problem dimension d. Using the theory of computational complexity, one can prove that *any* deterministic integration procedure is subject to the curse of dimensionality, at least in terms of a "worst case" measure of complexity.[19]

Since multivariate integration is a "subproblem" that must be solved in order to carry out dynamic programming when there are continuous state variables (indeed, dynamic programming in principle involves infinitely many integrals in order to calculate $EV_{t+1}(s,d)$, one for each possible value of (s,d)) if there is a curse of dimensionality associated with numerical integration of a single multivariate integral, then it should also not be surprising that dynamic programming is also subject to the same curse. There is also a a curse of dimensionality associated with global optimization of non-convex objective functions of continuous variables. Since optimization is also a subproblem of the overall dynamic programming problem,

[19]The curse of dimensionality can disappear if one is willing to adopt a Bayesian perspective and place a "prior distribution" over the space of possible integrands and consider an "average case" instead of a "worst case" notion of computational complexity.

this constitutes another reason why dynamic programming is subject to a curse of dimensionality. Under the standard worst case definition of computational complexity, Chow and Tsitsiklis (1989) proved that *no* deterministic algorithm can succeed in breaking the curse of dimensionality associated with a sufficiently broad class of dynamic programming problems with continuous state and decision variables. This negative result dashes the hopes of researchers dating back to Bellman and Dreyfus (1962) who conjectured that there might be sufficiently clever deterministic algorithms that can overcome the curse of dimensionality.[20]

However there are examples of *random algorithms* that can circumvent the curse of dimensionality. Monte carlo integration is a classic example. Consider approximating the (multidimensional) integral in equation (32) by using *random* quadrature abscissae $\{\tilde{a}_i\}$ that are N independent and identically distributed (*IID*) draws from the distribution $p_{t+1}(s'|s,d)$ and uniform quadrature weights equal to $w_i(s,d) = 1/N$. Then the law of large numbers and the central limit theorem imply that the Monte carlo integral $\hat{E}V_{t+1}(s,d)$ converges to the true conditional expectation $EV_{t+1}(s,d)$ at rate $1/\sqrt{N}$ *regardless of the dimension of the state space d*. Thus a random algorithm, Monte Carlo integration, succeeds in breaking the curse of dimensionality of multivariate integration.[21]

However naive application of Monte Carlo integration will not necessarily break the curse of dimensionality of the dynamic programming problem. The reason is that a form of *uniform convergence* (as opposed to pointwise) convergence of the conditional expectations $\hat{E}V_{t+1}(s,d)$ to $EV_{t+1}(s,d)$ is required in order to guarantee that the overall backward induction process converges to the true solution as the number of Monte Carlo draws, N, gets large.[22] Extending an idea introduced by Tauchen and Hussey (1991) to solve rational expectations models, Rust (1997) proved that it is possible to break the curse of dimensionality in a class of SDPs where the choice sets $D_t(s)$ are finite, a class he calls *disrete decision processes*.[23] The key idea is to choose, as a *random grid,* the same set of random points that are used quadrature abscissae for Monte Carlo integration. That is, suppose $p_{t+1}(s'|s,d)$ is a transition *density* and the state space (perhaps after translation and normalization) is identified with the d-dimensional *hypercube* $S = [0,1]^d$. Apply Monte carlo integration by drawing N *IID* points $\{\tilde{s}_1,\dots,\tilde{s}_N\}$ from the this hypercube (this can be accomplished by drawing each component of s_i from the uniform distribution on the $[0,1]$ interval). We have

$$\hat{E}V_{t+1}(s,d) = \frac{1}{N}\sum_{i=1}^{N} V_{t+1}(\tilde{s}_i)p_{t+1}(\tilde{s}_i|s,d). \tag{34}$$

Applying results from the theory of *empirical processes* (Pollard 1989), Rust showed that this form of the Monte Carlo integral does result in uniform convergence (i.e. $\|\hat{E}V_{t+1}(s,d) - EV_{t+1}(s,d)\| = O_p(1/\sqrt{N})$), and using this, he showed that this randomized version of backward induction succeeds in breaking the curse of dimensionality of the dynamic programming problem. The intuition of why this works is instead of trying to approximate the conditional expectation in (32) by computing *many independent Monte carlo integrals* (i.e. drawing separate sets of random abscissae $\{\tilde{a}_i\}$ from $p_{t+1}(s'|s,d)$ for each possible value

[20]It was clear that they were hopeful that the curse of dimensionality could be circumvented: "Let us now discuss one of the most promising techniques for overcoming the 'curse of dimensionality,' the approximation of functions by polynomials." Since polynomial approximation is a particular type of deterministic algorithm, it too is subject to the curse of dimensionality, a result that is well known in the literature on function approximation.

[21] Unfortunately, randomization does *not* succeed in breaking the curse of dimensionality associated with general nonconvex optimization problems with continuous multidimensional decision variables d. See Nemirovsky and Yudin (1983).

[22]To get an intuition why, note that if separate *IID* sets of quadrature abscissae $\{\tilde{a}_i\}$ where drawn for each (s,d) point that we wish to evaluate the Monte Carlo integral $\hat{E}V_{t+1}(s,d)$ at, the resulting function would be an extremely "choppy" and irregular function of (s,d) as a result of all the random variation in the various sets of quadrature abscissae.

[23]The restriction to finite choice sets is necessary, since as note in footnote 21, randomization does not succeed in breaking the curse of dimensionality of nonconvex optimization problems with continuous decision variables.

of (s,d)), the approach in equation (34) is to compute a *single Monte Carlo integral* where the random quadrature points $\{\tilde{s}_i\}$ are drawn from the uniform distribution on $[0,1]^d$, and the integrand is treated as the function $V_{t+1}(s')p_{t+1}(s'|s,d)$ instead of $V_{t+1}(s')$. The second important feature is that equation (34) has a *self-approximating* property: that is, since the quadrature abscissae are the same as the grid points at which we compute the value function, no auxiliary interpolation or function approximation is necessary in order to evaluate $\hat{E}V_{t+1}(s,d)$. In particular, if $p_{t+1}(s'|s,d)$ is a smooth function of s, then $\hat{E}V_{t+1}(s,d)$ will also be a smooth function of s. Thus, backward induction using this algorithm is extremely simple. Before starting backward induction we choose a value for N and draw N *IID* random vectors $\{\tilde{s}_1,\dots,\tilde{s}_N\}$ from the uniform distribution on the d-dimensional hypercube. This constitutes a random grid that remains fixed for the duration of the backward induction. Then we begin ordinary backward induction calculations, at each stage t computing $V_t(\tilde{s}_i)$ at each of the N random grid points, and using the self-approximating formula (34) to calculate the conditional expectation of the period $t+1$ value function using only the N stored values $(V_{t+1}(\tilde{s}_1),\dots,V_{t+1}(\tilde{s}_N))$ from the previous stage of the backward induction. See Keane and Wolpin (1994) for an alternative approach, which combines Monte Carlo integration with the curve fitting approaches discussed above. Note that the Keane and Wolpin approach will not generally succeed in breaking the curse of dimensionality since it requires approximation of functions of d variables which is also subject to a curse of dimensionality, as is well known from the literature on *nonparametric regression*.

There are other subclasses of SDPs for which it is possible to break the curse of dimensionality. For example, the family of *linear quadratic/Gaussian* (LQG) can be solved in polynomial time using highly efficient matrix methods, including efficient methods for solving the *matrix Ricatti equation* which is used to computer the *Kalman filter* for Bayesian LQG problems (e.g. problems where the agent only receives a noisy signal of a state variable of interest, and they update their beliefs about the unknown underlying state variable via Bayes rule).

Now consider stationary, infinite horizon Markovian decision problems. As noted in section 3.4, there is no "last" period from which to begin the backward induction process. However, if the utility function is time separable and discounted, then under fairly general conditions, it will be possible to approximate the solution arbitrarily closely by solving a finite horizon version of the problem, where the horizon T is chosen sufficiently large. As we noted in section 3.4, this is equivalent to solving for V, the fixed point to the contraction mapping $V=\Gamma(V)$ by the method of *successive approximations,* where Γ is the *Bellman operator* defined in equation (14) of section 3.4.

$$V_{t+1}=\Gamma(V_t). \tag{35}$$

Since successive approximations converges at a geometric rate, with errors satisfying the upper bound in equation (19), this method can converge at an unacceptably slow rate when the discount factor is close to 1. A more effective algorithm in such cases is *Newton's Method* whose iterates are given by

$$V_{t+1}=V_t-[I-\Gamma'(V_t)]^{-1}[V_t-\Gamma(V_t)], \tag{36}$$

where Γ' is the *Gateaux* or *directional derivative* of Γ, i.e. it is the linear operator given by

$$\Gamma'(V)(W)=\lim_{t\to 0}\frac{\Gamma(V+tW)-\Gamma(V)}{t}. \tag{37}$$

Newton's method converges *quadratically* independent of the value of the discount factor, as long as it is less than 1 (to guarantee the contraction property and the existence of a fixed point). In fact, Newton's method turns out to be equivalent to the method of *policy iteration* introduced by Howard (1960). Let δ

be any stationary decision rule, i.e. a candidate *policy*. Define the policy-specific conditional expectation operator E_δ by

$$E_\delta V(s) = \int_{s'} V(s')p(s'|s,\delta(s)). \tag{38}$$

Given a value function V_t, let δ_{t+1} be the decision rule implied by V_t, i.e.

$$\delta_{t+1}(s) = \underset{d\in D(s)}{argmax}\left[u(s,d)+\rho\beta\int_{s'} V_t(s')p(s'|s,d)\right]. \tag{39}$$

It is not hard to see that the value of policy δ_{t+1} must be at least as high as V_t, and for this reason, equation (39) is called the *policy improvement step* of the policy iteration algorithm. It is also not hard to show that

$$\Gamma'(V_t)(W)(s) = \rho\beta E_{\delta_{t+1}}W(s), \tag{40}$$

while implies that the Newton iteration, equation (36), is numerically identical to *policy iteration*

$$V_{t+1}(s) = [I-\rho\beta E_{\delta_{t+1}}]^{-1}u(s,\delta_{t+1}(s)), \tag{41}$$

where δ_{t+1} is given in equation (39). Equation (41) is called the *policy valuation step* of the policy iteration algorithm since it calculates the value function implied by the policy δ_{t+1}. Note that since E_δ is an expectation operator, it is linear and satisfies $\|E_\delta\| \leq 1$, and this implies that the operator $[I-\rho\beta E_\delta]$ is invertible and has the following geometric series expansion

$$[I-\rho\beta E_\delta]^{-1} = \sum_{j=0}^{\infty}[\rho\beta]^j E_\delta^j, \tag{42}$$

where E_δ^j is the *j step ahead expectations operator*. Thus, we see that

$$[I-\rho\beta E_\delta]^{-1}u(s,\delta(s)) = \sum_{j=0}^{\infty}[\rho\beta]^j E_\delta^j u(s,\delta(s)) = E\left\{\sum_{t=0}^{\infty}[\rho\beta]^t u(s_t,\delta(s_t))\middle| s_0=s\right\}, \tag{43}$$

so that value function V_t from the policy iteration (41) corresponds to the expected value implied by policy (decision rule) δ_t.

If there are an infinite number of states, the expectations operator E_δ is an infinite-dimensional linear operator, so it is not feasible to compute an exact solution to the policy iteration equation (41). However if there are a finite number of states (or an infinite state space is discretized to a finite set of points, as per the discussion above), then E_δ is an $N\times N$ transition probability matrix, and policy iteration is feasible using ordinary matrix algebra, requiring at most $O(N^3)$ operations to solve a system of linear equations for V_t at each policy valuation step. Further, when there are a finite number of possible actions as well as states, there are only a finite number of possible policies $|D|^{|S|}$, where $|D|$ is the number of possible actions and $|S|$ is the number of states, and policy iteration can be shown to converge in a finite number of steps, since the method produces an improving sequences of decision rules, i.e. $V_t \leq V_{t+1}$. Thus, since there is an upper bound on the number of possible policies and policy iteration cannot cycle, it must converge in a finite number of steps. The number of steps is typically quite small, far fewer than the total number of possible policies. Santos and Rust (2004) show that the number of iterations can be bounded independent of the number of elements in the state space, $|S|$. Thus, policy iteration is the method of choice for infinite horizon problems for which the discount factor is sufficiently close to 1. However if the discount factor is far enough below 1, then successive approximations can be faster since policy iteration requires $O(N^3)$ operations per iteration whereas successive approximations requires $O(N^2)$ operations per iteration. At most $T(\varepsilon,\beta)$ successive approximation iterations are required to compute an ε-approximation to an infinite horizon Markovian decision problem with discount factor β, where $T(\varepsilon,\beta) = \log((1-\beta)\varepsilon)/\log(\beta)$. Roughly speaking, if $T(\varepsilon,\beta) < N$, then successive approximations is faster than policy iteration.

Successive approximations can be accelerated by a number of means discussed in Puterman (1994) and Rust (1996). *Multigrid algorithms* are also effective: these methods begin backward induction with a coarse grid with relatively few grid points N, and then as iterations proceed, the number of grid points are successively increased leading to finer and finer grids as the backward induction starts to converge. Thus, computational time is not wasted early on in the backward induction iterations when the value function is far from the true solution. *Adaptive grid* methods are also highly effective in many problems: these methods can automatically detect regions in the state space where there is higher curvature in the value function, and in these regions more grid points are added in order to ensure that the value function is accurately approximated, whereas in regions where the value function is "flatter" grid points can be remove, so as to direct computational resources to the regions of the state space where there is the highest payoff in terms of accurately approximating the value function. See Grüne and Semmler (2004) for more details and an interesting application of adaptive grid algorithms.

I conclude this section with a discussion of several other alternative approaches to solving stationary infinite horizon problems that can be extremely effective relative to "discretization" methods when the number of grid points N required to obtain a good approximation becomes very large. Recall the curve fitting approach discussed above in finite horizon SDPs: we approximate the value function V by a parametric function as $V_\theta(s) \equiv f(s,\theta)$ for some flexible functional form f, where θ are treated as unknown parameters to be "estimated." For infinite horizon SDPs, our goal is to find parameter values $\hat{\theta}$ so that the implied value function satisfies the Bellman equation as well as possible. One approach to doing this, known as the *minimum residual method,* is a direct analog of nonlinear least squares: if θ is a vector with K components, we select $N \geq K$ points in the state space (potentially at random) and find $\hat{\theta}$ that minimizes the squared deviations or *residuals* in the Bellman equation

$$\hat{\theta} = \mathop{argmin}_{\theta \in R^K} \sum_{i=1}^{N} [\hat{\Gamma}(V_\theta)(s_i) - V_\theta(s_i)]^2, \tag{44}$$

where $\hat{\Gamma}$ denotes an approximation to the Bellman operator, where some numerical integration and optimization algorithm are used to approximate the true expectation operator and maximization in the Bellman equation (14). Another approach, called the *collocation method,* finds $\hat{\theta}$ by choosing K grid points in the state space and setting the residuals at those K points to zero:

$$\begin{aligned} V_{\hat{\theta}}(s_1) &= \hat{\Gamma}(V_{\hat{\theta}})(s_1) \\ V_{\hat{\theta}}(s_2) &= \hat{\Gamma}(V_{\hat{\theta}})(s_2) \\ &\cdots \\ V_{\hat{\theta}}(s_K) &= \hat{\Gamma}(V_{\hat{\theta}})(s_K). \end{aligned} \tag{45}$$

Another approach, called *parametric policy iteration* (Rust, *et. al.* 2004), carries out the policy iteration algorithm in equation (41) above, but instead of solving the linear system (41) for the value function V_t at each policy valuation step, they approximately solve this system by finding $\hat{\theta}_t$ that solves the regression problem

$$\hat{\theta}_t = \underset{\theta \in R^K}{argmin} \sum_{i=1}^{N} [V_{\hat{\theta}_t}(s_i) - u(s_i, \delta_t(s_i)) - \rho\beta E_{\delta_t} V_{\hat{\theta}_t}(s_i)]^2 \tag{46}$$

Other than this, policy iteration proceeds exactly as discussed above. Note that due to the linearity of the expectations operator, the regression problem above reduces to an ordinary linear regression problem when V_θ is approximated as a linear combination of basis functions as in (30) above.

There are variants the minimum residual and collocation methods that involve parameterizing the *decision rule* rather than the value function. These methods are frequently used in problems where the control variable is continuous, and construct residuals from the *Euler equation* — a functional equation for the decision rule that can in certain classes of problems be derived from the first order necessary condition for the optimal decision rule. These approaches then try to find $\hat{\theta}$ so that the Euler equation (as opposed to the Bellman equation) is approximately satisfied, in the sense of minimizing the squared residuals (minimum residual approach) or setting the residuals to zero at K specified points in the state space (collocation method). See Judd (1998) for further discussion of these methods and a discussion of strategies for choosing the grid points necessary to implement the collocation or minimum residual method.

There are variety of other *iterative stochastic algorithms* for approximating solutions to dynamic programming problems that have been developed in the computer science and "artificial intelligence" literatures on *reinforcement learning*. These methods include *Q-learning, temporal difference learning,* and *real time dynamic programming*. The general approach in all of these methods is to iteratively update an estimate of the value function, and recursive versions of Monte Carlo integration methods are employed in order to avoid doing numerical integrations to calculate conditional expectations. Using methods adapted from the literature on *stochastic approximation* it is possible to prove that these methods converge to the true value function in the limit as the number of iterations tends to infinity. A key assumption underlying the convergence proofs is that there is sufficient stochastic noise to assure that all possible decisions and decision nodes are visited "infinitely often." The intuition of why such an assumption is necessary follows from the discussion in section 3: suppose that at some state s an initial estimate of the value function for decision that is actually optimal happens to be so low that the action is deemed to be "nonoptimal" relative to the initial estimate. If the agent does not "experiment" sufficiently, and thus fails to choose sub-optimal decisions infinitely often, the agent may fail to learn that the initial estimated value was an under-estimate of the true value, and therefore the agent might never learn that the corresponding action really is optimal.[24] A nice feature of many of these methods, particularly the real time dynamic programming developed in Barto, Bradtke and Singh (1995), is that these methods can be used in "real time", i.e. we do not have to "precalculate" the optimal decision rule in "offline" mode. All of these algorithms result in steady improvement in performance with experience. Methods similar to these have been used to produce highly effective strategies in extremely complicated problems. An example isIBM's "Deep Blue" computer chess strategy, which has succeeded in beating the world's top human chess players (e.g. Garry Kasparov). However the level of computation and repetition necessary to "train" effective strategies is huge and time consuming, and it is not clear that any of these methods succeed in breaking the curse of dimensionality. For further details on this literature, see Bertsekas and Tsitsiklis (1996). Pakes (2001)

[24]There is a trade-off between learning and experimentation, of course. The literature on "multi-armed bandits" (Gittins, 1979) shows that a fully rational Bayesian decision maker will generally not find it optimal to experiment infinitely often. As a result such an agent can fail to discover actions that are optimal in an *ex post* sense. However this does not contradict the fact that their behavior is optimal in an *ex ante* sense: rather it is a reflection that learning and experimentation is a costly activity, and thus it can be optimal to be incompletely informed, a result that has been known as early as Wald (1947).

applies these methods to approximate Markov perfect equilibria in games with many players. All types of stochastic algorithms have the disadvantage that the approximate solutions can be "jagged" and there is always at least a small probability that the converged solution can be far from the true solution. However they may be the only feasible option in many complex, high-dimensional problems where deterministic algorithms (e.g. the Pakes-McGuire (1994) algorithm for Markov perfect equilibrium) quickly become intractable due to the curse of dimensionality.

5 Empirical Dynamic Programming and The Identification Problem

The developments in numerical dynamic programming described in the previous section paved the way for a new, rapidly growing literature on empirical estimation and testing of SDPs and dynamic games. This literature began to take shape in the late 1970s, with contributions by Sargent (1978) on estimation of dynamic labor demand schedules in a linear quadratic framework, and Hansen and Singleton (1982) who developed a generalized method of moment estimation strategy for a class of continuous choice SDPs using the *Euler equation* as an *orthogonality condition*. About the same time, a number of papers appeared that provided different strategies for estimation and inference in *dynamic discrete choice models* including Gotz and McCall's (1980) model of retirements of air force pilots, Wolpin's (1984) model of a family's decision whether or not to have a child, Pakes's (1986) model of whether or not to renew a patent, and Rust's (1987) model of whether or not to replace a bus engine. In the last 20 years, hundreds of different empirical applications of dynamic programming models have been published. For surveys of this literature see Ekstein and Wolpin (1989), Rust (1994), and the very readable book by Adda and Cooper (2003) (which also provides accessible introductions to the theory and numerical methods for dynamic programming). The remainder of this section will provide a brief overview of estimation methods and a discussion of the identification problem.

In econometrics, the term *structural estimation* refers to a class of methods that try to go beyond simply summarizing the behavior of economic agents by attempting to infer their underlying *preferences* and *beliefs*. This is closely related to the distinction between the *reduced-form* of an economic model and the underlying *structure* that "generates" it.[25] The reason why one would want to do structural estimation, which is typically far more difficult (e.g. computationally intensive) than reduced-form estimation, is having knowledge of underlying structure enables us to conduct *hypothetical/counterfactual policy experiments*. Reduced-form estimation methods can be quite useful and yield significant insights into behavior, but they are limited to summarizing behavior under the *status quo*. However they are unable to forecast how individuals change their behavior in response to various changes in the environment, or in *policies* (e.g. tax rates, government benefits, regulations, laws, etc.) that *change the underlying structure* of agents decision problems. As long as it is possible to predict how different policies change the underlying structure, we can use dynamic programming to re-solve agents' SDPs under the alternative structure, resulting in corresponding decision rules that represent predictions of how their behavior (and welfare) will change in response to the policy change.

The rationale for structural estimation was recognized as early as Marschak (1953), however his message appears to have been forgotten until the issue was revived in Lucas's (1976) critique of the limitations of reduced-form methods for policy evaluation. An alternative way to do policy evaluation is via *random-*

[25]Structural estimation methods were first developed at the Cowles Commission at Yale University, starting with attempts to structurally estimate the linear simultaneous equations model, and models of investment by firms. Frisch, Haavelmo, Koopmans, Marschak, and Tinbergen were among the earliest contributors to this literature.

ized experiments in which subjects are randomly assigned to the *treatment group* (where the "treatment" is some alternative policy of interest) and the *control group* (who continue with the policy under the *status quo*). By comparing the outcomes in the treatment and control groups, we can assess the behavioral and welfare impacts of the policy change. However human experiments can be very time consuming and expensive to carry out, whereas "computational experiments" using a structural model are very cheap and be conducted extremely rapidly. However the drawback of the structural approach is the issue of *credibility* of the structural model. If the structural model is *misspecified* it can generate incorrect forecasts of the impact of a policy change. There are numerous examples of how structural models can be used to make policy predictions. See Todd and Wolpin (2005) for an example that compares the prediction of a structural model with the results of a randomized experiment, where the structural model is estimate using subjects from the control group, and *out of sample predictions* are made to predict the behavioral response by subjects in the treatment group. They show that the structural model results in accurate predictions of how the treatment group subjects responded to the policy change.

I illustrate the main econometric methods for structural estimation of SDPs in the case of a stationary infinite horizon Markovian decision problem, although all the concepts extend in a straightforward fashion to finite horizon, non-stationary and non-Markovian problems. Estimation requires a specification of the *data generating process*. Assume we observe N agents, and we observe agent i from time period $\underline{T}_i$ to $\overline{T}_i$, (or via appropriate re-indexing, from $t = 1,\ldots,T_i$). Assume observations of each individual are independently distributed realizations from the controlled process $\{s_t, d_t\}$. However while we assume that we can observe the decisions made by each agent, it is more realistic to assume that we only observe a subset of the agent's state s_t. If we partition $s_t = (x_t, \varepsilon_t)$, assume that the econometrician observes x_t but not ε_t, so this latter component of the state vector constitutes an *unobserved state variable*. Then the reduced-form of the SDP is the decision rule δ

$$d = \delta(x, \varepsilon), \tag{47}$$

since the decision rule embodies all of the behavioral content of the SDP model. The *structure* Λ consists of the objects $\Lambda = \{\beta, \rho, u(s,d), p(s'|s,d)\}$. Equation (12) specifies the mapping from the structure Λ into the reduced form, δ. The data set consists of $\{(x_{i,t}, d_{i,t}), t = 1,\ldots,T_i,\ i = 1,\ldots,N\}$. The econometric problem is to infer the underlying structure Λ from our data on the observed states and decisions by a set of individuals. Although the decision rule is potentially a complicated nonlinear function of unobserved state variables in the reduced-form equation (47), it is often possible to consistently estimate the decision rule under weak assumptions as $N \to \infty$, or as $T_i \to \infty$ if the data consists only of a single agent or a small number of agents i who are observed over long intervals. Thus, the decision rule δ can be treated as a *known function* for purposes of a theoretical analysis of identification. The *identification problem* is the question, *under what conditions is the mapping from the underlying structure* Λ *to the reduced form* δ 1 *to* 1 *(i.e. invertible)?* If this mapping is 1 to 1, we say that the structure is *identified* since in principle it can be inverted to uniquely determine the underlying structure Λ. In practice, we construct an *estimator* $\hat{\Lambda}$ based on the available data and show that $\hat{\Lambda}$ converges to the true underlying structure Λ as $N \to \infty$ and/or $T_i \to \infty$ for each i.

Unfortunately, rather strong *a priori* assumptions on the form of agents' preferences and beliefs are required in order to guarantee identification of the structural model. Rust (1994) and Magnac and Thesmar (2002) have shown that an important subclass of SDPs, *discrete decision processes* (DDPs), are *non-parametrically unidentified*. That is, if we are unwilling to make any *parametric* functional form assumptions about preferences or beliefs, then in general there are infinitely many different structures Λ consistent with any reduced form δ. In more direct terms, there are many different ways to *rationalize* any observed pattern of behavior as being "optimal" for different configurations of preferences and beliefs. It is likely that these results extend to continuous choice problems, since it is possible to approximate a continuous decision process (CDP) by a sequence of DDPs with expanding numbers of elements in their choice sets. Further, for dynamic games, Ledyard (1986), has shown that *any* undominated strategy profile can be a Bayesian equilibrium for some set of preferences and beliefs. Thus, the hypothesis of optimality or equilibrium *per se* does not have testable empirical content: further *a priori* assumptions must be imposed in order for SDPs models to be identified and result in empirically testable restrictions on behavior.

There are two main types of identifying assumptions that have been made in the literature to date: 1) parametric functional form assumptions on preferences $u(s,d)$ and components of agents' beliefs $p(s'|s,d)$ that involve unobserved state variables ε, and 2) *rational expectations*. Rational expectations states that an agent's *subjective beliefs* $p(s'|s,d)$ coincide with *objective probabilities* that can be estimated from data. Of course, this restriction is useful only for those components of s, x, that the econometrician can actually observe. In addition, there are other more general *functional restrictions* that can be imposed to help identify the model. One example is monotonicity and shape restrictions on preferences (e.g. concavity and monotonicity of the utility function), and another example are independence or *conditional independence* assumptions about variables entering agents' beliefs. I will provide specific examples below, however it should be immediately clear why these additional assumptions are necessary.

For example, consider the two parameters ρ (the agent's subjective survival probability) and β (the agent's subjective discount factor). We have seen in section 3 that *only the product of* ρ *and* β *enter the SDP model, and not* ρ *and* β *separately.* Thus, at most the product $\rho\beta$ can be identified, but without further assumptions it is impossible to separately identify the subjective survival probability ρ from the subjective discount factor β since both affect an agents' behavior symmetrical fashion. However we can separately identify ρ and β if we assume that an individual has *rational survival expectations,* i.e. that their subjective survival probability ρ coincides with the "objective" survival probability. Then we can estimate ρ "outside" the SDP model, using data on the lifetime distributions of similar types of agents, and then β can be identified if other restrictions are imposed to guarantee that the product $\rho\beta$ is identified. However it can very difficult to make precise inferences about agents' discount factors in many problems, and it is easy to think of models where there is heterogeneity in survival probabilities and discount factors, and unobserved variables affecting one's beliefs about them (e.g. family characteristics such as a predisposition for cancer, etc. that are observed by an agent but not by the econometrician) where identification is problematic.

There are two main approaches for conducting inference in SDPs: 1) maximum likelihood, and 2) "simulation estimation". The latter category includes a variety of similar methods such as *indirect inference* (Gourieroux and Monfort, 1997), *simulated method of moments* (McFadden, 1989, Gallant and Tauchen, 1996), *simulated maximum likelihood and method of simulated scores* (Ruud, 2006), and *simulated minimum distance* (Hall and Rust, 2006). To simplify the discussion I will define these initially for single agent SDPs and at the end discuss how these concepts naturally extend to dynamic games. I will illustrate maximum likelihood and show how a likelihood can be derived for a class of DDPs, however for CDPs, it is typically much more difficult to derive a likelihood function, especially when there are issues of *censoring,* or problems involving mixed discrete and continuous choice. In such cases simulation estimation is often the only feasible way to do inference.

For discrete decision processes, assume that the utility function has the following parametric, *additively separable* representation

$$u(x,\varepsilon,d) = u(x,d,\theta_1) + \varepsilon(d) \quad \text{(AS)}. \tag{48}$$

where $\varepsilon = \{\varepsilon(d)|d \in D(x)\}$, and $\varepsilon(d)$ is interpreted as an unobserved component of utility associated with choice of alternative $d \in D(x)$. Further, suppose that the transition density $p(x',\varepsilon'|x,\varepsilon,d)$ satisfies the following *conditional independence assumption*

$$p(x',\varepsilon'|x,\varepsilon,d) = p(x'|x,d,\theta_2)q(\varepsilon',\theta_3) \quad \text{(CI)}. \tag{49}$$

The CI assumption implies that $\{\varepsilon_t\}$ is an *IID* "noise" process that is independent of $\{x_t,d_t\}$. Thus all of the serially correlated dynamics in the state variables are captured by the observed component of the state vector x_t. If in addition $q(\varepsilon_t,\theta_3)$ is a distribution with unbounded support with finite absolute first moments, One can show that the following *conditional choice probabilities* exist

$$P(d|x,\theta) = \int I_\varepsilon\{d = \delta(x,\varepsilon,\theta)\}\, q(\varepsilon)d\varepsilon, \tag{50}$$

where $\theta = (\rho,\beta,\theta_1,\theta_2,\theta_3)$ constitute the vector of unknown parameters to be estimated.[26] In general, the parametric functional form assumptions, combined with the assumption of rational expectations and the AS and CI assumptions are sufficient to identify the unknown parameter vector θ^*. θ^* can be estimated by maximum likelihood, using the *full information* likelihood function $\mathcal{L}_f$ given by

$$\mathcal{L}_f(\theta|\{x_{i,t},d_{i,t}\},t=1,\ldots,T_i,\ i=1,\ldots,N) = \prod_{i=1}^{N}\prod_{t=2}^{T_i} P(d_{i,t}|x_{i,t},\theta)p(x_{i,t}|x_{i,t-1},d_{i,t-1},\theta_2). \tag{51}$$

A particularly tractable special case is where $q(\varepsilon,\theta_3)$ has a *multivariate extreme value distribution* where θ_3 is a common scale parameter (linearly related to the standard deviation) for each variable in this distribution (see Rust (2006) and Magnac (2006) for the exact formula for this density). This specification leads to a dynamic generalization of the *multinomial logit model*

$$P(d|x,\theta) = \frac{\exp\{v(x,d,\theta)/\theta_3\}}{\sum_{d'\in D(x)}\exp\{v(x,d',\theta)/\theta_3\}}, \tag{52}$$

where $v(x,d,\theta)$ is the expected, discounted utility from taking action d in observed state x given by the unique fixed point to the following *smoothed Bellman equation*

$$v(x,d,\theta) = u(x,d,\theta_1) + \rho\beta\int_{x'}\theta_3\log\left(\sum_{d'\in D(x')}\exp\{v(x',d',\theta)/\theta_3\}\right)p(x'|x,d,\theta_2)dx'. \tag{53}$$

Define Γ_θ by

$$\Gamma_\theta(W)(x,d) = u(x,d,\theta_1) + \rho\beta\int_{x'}\theta_3\log\left(\sum_{d'\in D(x')}\exp\{\exp\{W(x',d',\theta)/\theta_3\}\right)p(x'|x,d,\theta_2)dx'. \tag{54}$$

It is not hard to show that under weak assumptions Γ_θ is a contraction mapping, so that $v(x,d,\theta)$ exists and is unique. Maximum likelihood estimation can be carried out using a *nested fixed point* maximum likelihood algorithm consisting of an "outer" optimization algorithm to search for a value of θ that maximizes $\mathcal{L}_f(\theta)$, and an "inner" fixed point algorithm that computes $v_\theta = \Gamma_\theta(v_\theta)$ each time the outer optimization

[26]Identification of fully parametric models is a "generic" property, that is, if there are two different parameters θ that produce the same conditional choice probability $P(d|x,\theta)$ for all x and $d \in D(x)$ (and thus lead to the same limiting expected log-likelihood), small perturbations in the parameterization will "almost always" result in a nearby model for which θ is uniquely identified.

algorithm generates a new trial guess for θ.[27] See Rust (1988) for further details on the nested fixed point algorithm and the properties of the maximum likelihood estimator, and Rust (1994) for a survey of alternative less efficient but computationally simpler estimation strategies.

As noted above, econometric methods for CDPs, i.e. problems where the decision variable is continuous (such as firm investment decisions, price settings, or consumption/savings decisions) are harder, since there is no tractable, general specification for the way unobservable state variables to enter the decision rule that that result in a *nondegenerate* likelihood function (i.e. where the likelihood $\mathcal{L}(\theta)$ is non zero for any data set and any value of θ). For this reason maximum likelihood estimation of CDPs is rare, outside of certain special subclasses, such at linear quadratic CDPs (Hansen and Sargent 1980 and Sargent 1981). However simulation based methods of inference can be used in a huge variety of situations where a likelihood is difficult or impossible to derive. These methods have a great deal of flexibility, a high degree of generality, and often permit substantial computational savings. In particular, generalizations of McFadden's (1989) *method of simulated moments* (MSM) have enabled estimation of a wide range of CDPs. The MSM estimator minimizes a quadratic form between a set of moments constructed from the data, h_N and a vector of *simulated moments* $h_{N,S}(\theta)$, i.e.

$$\begin{aligned} h_N &= \frac{1}{N}\sum_{i=1}^{N} h(\{x_{it}, d_{it}\}) \\ h_{N,S}(\theta) &= \frac{1}{S}\sum_{j=1}^{S}\frac{1}{N}\sum_{i=1}^{N} h(\{\tilde{x}^j_{it}(\theta), \tilde{d}^j_{it}(\theta)\}) \end{aligned} \tag{55}$$

where h is a vector of $J \geq K$ "moments" (i.e. functionals of the data that the econometrician is trying to "match"), where K is the dimension of θ, $\{x_{it}, d_{it}\}$ are the data, and $\{\tilde{x}^j_{it}(\theta), \tilde{d}^j_{it}(\theta)\}$, $j = 1, \ldots, S$ are S *IID* realizations of the controlled process. The estimate $\hat{\theta}$ is given by

$$\hat{\theta} = \underset{\theta \in R^K}{argmin}[h_N - h_{N,S}(\theta)]' W_N [h_N - h_{N,S}(\theta)], \tag{56}$$

where W_N is a $J \times J$ positive-definite weighting matrix. The most efficient choice for W_N is $W_N = [\hat{\Omega}_N]^{-1}$ where $\hat{\Omega}_N$ is the variance-covariance matrix formed from the vector of sample moments h_N. Simulation estimators require a nested fixed point algorithm since each time the outer minimization algorithm tries a new trial value for θ, the inner fixed point problem must be called to solve the CDP problem, using the optimal decision rule $d^j_{it}(\theta) = \delta(x^j_{it}, \varepsilon^j_{it}, \theta)$ to generate the simulated decisions, and the transition density $p(x^j_{i,t+1}, \varepsilon^j_{i,t+1} | x^j_{i,t}, \varepsilon^j_{i,t}, d^j_{i,t}, \theta_2)$ to generate $j = 1, \ldots, S$ *IID* realizations for a simulated panel each potential value of θ.[28]

Simulation methods are extremely flexible for dealing with a number of data issues such as attrition, missing data, censoring, and so forth. The idea is that if we are willing to build a stochastic model of the data "problem," we can account for it in the process of simulating the behavioral model. For example Hall and Rust (2006) develop a dynamic model of commodity price speculation in the steel market. An object of interest is to estimate the stochastic process governing wholesale steel prices, however there is

[27] The implicit function theorem guarantees that v_θ is a smooth function of θ. See Aguirregabiria and Mira (2004) for an ingenious alternative that "swaps" the order of the inner and outer algorithms of the nested fixed point algorithm resulting in significant computational speedups.

[28] It is important to simulate using a "common random numbers" that remain fixed as θ varies over the course of the estimation, in order to satisfy the *stochastic equicontinuity conditions* necessary to establish consistency and asymptotic normality of the simulation estimator.

no public commodity market where steel is traded and prices are recorded on a daily basis. Instead, Hall and Rust observe only the actual wholesale prices of a particular steel trader, who only records wholesale prices on the days he actually buys steel in the wholesale market. Since the speculator makes money by "buying low and selling high" the set of observed wholesale prices are *endogenously sampled,* and failure to account for this can lead to incorrect inferences about wholesale prices — a dynamic analog of *sample selection bias*. However in a simulation model it is easy to censor the simulated data the same way it is censored in the actual data, i.e. by discarding simulated wholesale prices on days where no simulated purchases are made. Hall and Rust show that even though moments based on the observed (censored) data are "biased" estimates, the simulated moments are biased in exactly the same fashion, but minimizing the distance between actual and simulated biased moments nevertheless results in consistent and asymptotically normal estimates of the parameters of the wholesale price process and other parameters entering the speculator's objective function.

The most recent literature has extended the methods for estimation of single agent SDPs to multi-agent dynamic games. For example, Rust (1994) described of dynamic discrete choice models to multiple agent *discrete dynamic games*. The unobserved state variables ε_t entering any particular agent's payoff function are assumed to be unobserved both by the econometrician and by the other players in the game. The *Bayesian Nash equilibria* of this game can be represented as a vector of conditional choice probabilities $(P_1(d_1|x),\dots,P_n(d_n|x))$, one for each player, where $P_i(d_i|x)$ represents the econometrician's and the other players beliefs about the probability player i will take action d_i, "integrating out" over the unobservable states variable $\varepsilon_{i,t}$ affecting player i's decision at time t similar to equation (50) for single agent problems. Adapting the numerical methods for Markov-perfect equilibrium described in section 4, it is possible to compute Bayesian Nash equilibria of discrete dynamic games using nested fixed point algorithms. While it is relatively straightforward to write down the likelihood function for the game, actual estimation via a straightforward application of full information maximum likelihood is extremely computationally demanding since it requires a *doubly nested fixed point algorithm* (i.e. an "outer" algorithm to search over θ to maximize the likelihood, and then an inner algorithm to solve the dynamic game for each value of θ, but this inner algorithm is itself a nested fixed point algorithm). Alternative less computationally demanding estimation methods have been proposed by Aguirregabiria and Mira (2006), Bajari and Hong (2006), and Pesendorfer and Schmidt-Dengler (2003). This research is at the current frontier of development in numerical and empirical applications of dynamic programming.

Besides econometric methods, which are applied for structural estimation for actual agents in their "natural" settings, an alternative approach is to try to make inferences about agents' preferences and beliefs (and even their "mode of reasoning") for artificial SDPs in a laboratory setting. The advantage of a laboratory experiment is *experimental control over preferences and beliefs*. The ability to control these aspects of decision making can enable much tighter tests of theories of decision making. For example, the *Allais paradox* is a classic example of a simple laboratory experiment that casts doubt on the hypothesis that humans are expected utility maximizers. Other experiments have been constructed to determine if humans actually use backward induction to solve complicated dynamic decision problems. For example, Binmore *et. al*. (2002) structured a laboratory experiment to determine whether individuals do backward induction in one and two stage alternating offer games, and "find systematic violations of backward induction that cannot be explained by payoff-interdependent preferences." (p. 49).

6 Comments

There has been tremendous growth in research related to dynamic programming in the six decades. The method has evolved into the main tool for solving sequential decision problems, and research related to dynamic programming has lead to fundamental advances in theory, numerical methods, and econometrics. As we have seen, while dynamic programming embodies the notion of rational decision making under uncertainty, there is mixed evidence as to whether it provides a good literal description of how human beings actually behave in comparable situations. Although human reasoning and decision making is undoubtedly both more complex and more "frail" and subject to foibles and limitations than the idealized notion of "full rationality" that dynamic programming embodies, the discussion of the identification problem shows that if we are given sufficient flexibility about how to model individual preferences and beliefs, there exist SDPs whose decision rules provide arbitrarily good approximations to individual behavior.

Thus, dynamic programming can be seen as a useful "first approximation" to human decision making, but it will undoubtedly be superseded by more descriptively accurate psychological models. Indeed, in the future one can imagine behavioral models that are not derived from some *a priori* axiomatization of preferences, but will result from empirical research that will ultimately deduce human behavior from yet even deeper "structure" i.e. the very underlying neuroanatomy of the human brain.

Even if dynamic programming will be superseded as a model of human behavior, it will likely still remain highly relevant for the foreseeable future as the embodiment of rational behavior. Indeed some "futurists" (e.g. Kurzweil, 2005) predict that in the not too distant future (e.g. approximately 2050) a *singularity* will occur, "during which the pace of technological change will be so rapid, its impact so deep, that human life will be irreversibly transformed." (p. 7). Kurzweil and others foresee very rapid developments in "artificial intelligence" that will overcome many of the limitations of the human brain and human reasoning: "By the end of this century, the nonbiological portion of our intelligence will be trillions and trillions of times more powerful than unaided human intelligence." (p. 9). Whether this prognosis will ever come to pass, or come to pass as soon as Kurzweil forecast is debatable. However it does suggest that there will be continued interest and research on dynamic programming.

However the fact that reasonably broad classes of dynamic programming problems are subject to a curse of dimensionality suggests that it may be too optimistic to think that human rationality will be superseded by "artificial rationality" any time soon. While there are many complicated problems that we would like to solve by dynamic programming in order to understand what "fully rational" behavior actually looks like in specific situations, the curse of dimensionality still limits us to very simple "toy models" that only very partially and simplistically capture the myriad of details and complexities we face in the real world. Although we now have a number of examples where artificial intelligence based on principles from dynamic programming outstrips human intelligence, e.g. computerized chess, all of these cases are for very specific problems in very narrow domains. I believe that it will be a long time before technological progress in computation and algorithms produce truly general purpose "intelligent behavior" that can compete successfully with human intelligence in widely varying domains and in the immensely complicated situations that we operate in every day. Despite all of our psychological frailties and limitations, there is an important unanswered question of "how do we do it?" and more research is required to determine if human behavior is simply suboptimal, or whether the human brain uses some powerful implicit "algorithm" to circumvent the curse of dimensionality that digital computers appear to be subject to for solving problems such as SDPs by dynamic programming.[29]

[29]For a provocative theory that deep principles of quantum mechanics can enable human intelligence to transcend computational limitations of digital computers, see Penrose (1989).

References

[1] Abreu, D., D. Pearce and E. Stacchetti (1986) "Optimal Cartel Equilibria with Imperfect Monitoring" *Journal of Economic Theory* **39** 251–269.

[2] Abreu, D. D. Pearce and E. Stacchetti (1990) "Toward a Theory of Discounted Repeated Games with Imperfect Monitoring" *Econometrica* **58** 1041–1063.

[3] Adda, J. and R. Cooper (2003) *Dynamic Economics Quantitative Methods and Applications* MIT Press, Cambridge, Massachusetts.

[4] Aguirregabiria, V. and P. Mira (2004) "Swapping the Nested Fixed Point Algorithm: A Class of Estimators for Discrete Markov Decision Models" *Econometrica* **70** 1519–1543.

[5] Aguirregabiria, V. and P. Mira (2005) "Sequential Estimation of Dynamic Discrete Games" manuscript, University of Toronto.

[6] Arrow, K.J., D. Blackwell, and M.A. Girshik (1949) "Bayes and Minimax Solutions of Sequential Decision Problems" *Econometrica* **17** 213–244.

[7] Arrow, K.J., T. Harris and J. Marschak (1951) "Optimal Inventory Policy" *Econometrica* **19** 250–272.

[8] Bajari, P. and H. Hong (2004) "Semiparametric Estimation of a Dynamic Game of Incomplete Information" manuscript, Duke University.

[9] Barto, A. G., S. J. Bradtke and S. P. Singh (1995) "Learning to Act Using Real-Time Dynamic Programming" *Artificial Intelligence* **72** 81–138.

[10] Bellman, R. (1957) *Dynamic Programming* Princeton University Press, Princeton, New Jersey.

[11] Bellman, R. and S. Dreyfus (1962) *Applied Dynamic Programming* Princeton University Press, Princeton, New Jersey.

[12] Bellman, R. (1984) *Eye of the Hurricane* World Scientific Publishing Company, Singapore.

[13] Bertsekas, D. P. (1995) *Dynamic Programming and Optimal Control, volumes 1 and 2* Athena Scientific, Belmont, Massachusetts.

[14] Bertesekas, D.P. and J. Tsitsiklis (1996) *Neuro-Dynamic Programming* Athena Scientific, Belmont, Massachusetts.

[15] Bhattacharya, R.N. Majumdar, M. (1989) "Controlled Semi-Markov Models – The Discounted Case" *Journal of Statistical Planning and Inference* **21** 365–381.

[16] Binmore, K, J. McCarthy, G. Ponti, L. Samuelson and A. Shaked (2002) "A Backward Induction Experiment" *Journal of Economic Theory* **104** 48–88.

[17] Blackwell, D. (1962) "Discrete Dynamic Programming" *Annals of Mathematical Statistics* **33** 719–726.

[18] Blackwell, D. (1965) "Positive Dynamic Programming" *Proceedings of the 5th Berkeley Symposium* **3** 415–428.

[19] Blackwell, D. (1965) “Discounted Dynamic Programming” *Annals of Mathematical Statistics* **36** 226–235.

[20] Chow, C.S. and Tsitsiklis, J.N. (1989) “The Complexity of Dynamic Programming” *Journal of Complexity* **5** 466–488.

[21] Denardo, E. (1967) “Contraction Mappings Underlying the Theory of Dynamic Programming” *SIAM Review* **9** 165–177.

[22] Eckstein, Z. and K. I. Wolpin (1989) “The Specification and Estimation of Dynamic Stochastic Discrete Choice Models: A Survey” *Journal of Human Resources* **24** 562–598.

[23] Epstein, L. and S. Zin (1989) “Substitution, Risk Aversion, and the Temporal Behavior of Consumption and Asset Returns: A Theoretical Framework” *Econometrica* **57** 937–969.

[24] Epstein, L. and M. Le Breton (1993) “Dynamically Consistent Beliefs Must be Bayesian” *Journal of Economic Theory* **61** 1-22.

[25] Epstein, L. and M. Schneider (2003) “Recursive Multiple Priors” *Journal of Economic Theory* **113** 1–31.

[26] Gallant, A.R. and G.E. Tauchen (1996) “Which Moments to Match?” *Econometric Theory* **12** 657–681.

[27] Gihman, I.I. and A.V. Skorohod (1979) *Controlled Stochastic Processes* Springer-Verlag, New York.

[28] Gittins, J. C. (1979) “Bandit Processes and Dynamic Allocation Indices” *Journal of the Royal Statistical Society, volume B* **41** 148–164.

[29] Gotz, G.A. McCall, J.J. (1980) “Estimation in Sequential Decisionmaking Models: A Methodological Note” *Economics Letters* **6** 131–136.

[30] Gourieroux, C. and A. Monfort (1997) *Simulation-Based Methods of Inference* Oxford University Press, Oxford, UK.

[31] Grüne, L. and W. Semmler (2004) “Using dynamic programming with adaptive grid scheme for optimal control problems in economics.” **28** 2427–2456.

[32] Gul, F. and W. Pesendorfer (2004) “Self-Control and the Theory of Consumption” *Econometrica* **72** 119–158.

[33] Hall, G. and J. Rust (2006) “Econometric Methods for Endogenously Sampled Time Series: The Case of Commodity Price Speculation in the Steel Market” manuscript, Yale University.

[34] Hammond, P. (1988) “Consequentalist Foundations for Expected Utility” *Theory and Decision* **25** 25–78.

[35] Hansen, L.P. and T.J. Sargent (1980) “Formulating and Estimating Dynamic Linear Rational Expectations Models” *Journal of Economic Dynamics and Control* **2** 7–46.

[36] Hansen, L.P. Singleton, K. (1982) “Generalized Instrumental Variables Estimation of Nonlinear Rational Expectations Models” *Econometrica* **50** 1269–1281.

[37] Howard, R.A. (1960) *Dynamic Programming and Markov Processes* John Wiley and Sons, New York.

[38] Judd, K. (1998) *Numerical Methods in Economics* MIT Press, Cambridge, Massachusetts.

[39] Judd, K., S. Yeltekin, and J. Conklin (2003) "Computing Supergame Equilibria" *Econometrica* **71** 1239–1254.

[40] Keane, M. and K.I. Wolpin (1994) "The Solution and Estimation of Discrete Choice Dynamic Programming Models by Simulation: Monte Carlo Evidence" *Review of Economics and Statistics* **76** 648–672.

[41] Kirby, K, and R. J. Herrnstein (1995) "Preference Reversal due to Myopia of Delayed Reward" *Psychological Science* 83–95.

[42] Koopmans, T. (1960) "Stationary Ordinal Utility and Impatience" *Econometrica* **28** 287–309.

[43] Krusell, P. and A. Smith (1998) "Income and Wealth Heterogeneity in the Macroeconomy" *Journal of Political Economy* **106** 867–896.

[44] Kurzweil, R. (2005) *The Singularity is Near When Humans Transcend Biology* Viking Press, New York.

[45] Kushner, H.J. (1990) "Numerical Methods for Stochastic Control Problems in Continuous Time" *SIAM Journal on Control and Optimization* **28** 999–1048.

[46] Ledyard, J. (1986) "The scope of the hypothesis of Bayesian equilibrium" *Journal of Economic Theory* **39** 59–82.

[47] Lucas, R.E. Jr. (1976) "Econometric Policy Evaluation: A Critique" in K. Brunner and A.K Meltzer (eds.) *The Phillips Curve and Labour Markets* Carnegie-Rohcester Conference on Public Policy, North Holland.

[48] Lucas, R.E. Jr. (1978) "Asset Prices in an Exchange Economy" *Econometrica* **46** 1426–1445.

[49] Ljungqvist, L. and T. J. Sargent *Recursive Macroeconomic Theory* MIT Press, Cambridge, Massachusetts.

[50] Machina, M. (1989) "Dynamic Consistency and Non-Expected Utility Models of Choice under Uncertainty" *Journal of Economic Literature* **27** 1622-1668.

[51] Magnac, T. and D. Thesmar (2002) "Identifying Dynamic Discrete Decision Processes" *Econometrica* **70** 801–816.

[52] Marschak, T. (1953) "Economic Measurements for Policy and Prediction" in W.C. Hood and T.J. Koopmans (eds.) *Studies in Econometric Method* Wiley, New York.

[53] Maskin, E. and J. Tirole (2001) "Markov Perfect Equilibrium I. Observable Actions" *Journal of Economic Theory* **100** 191-219.

[54] Massé, P. (1945) "Application des Probabilités en Chaîne á l'hydrologie statistique et au jeu des Résevoirs" Report to the Statistical Society of Paris, Berger-Levrault, Paris.

[55] Massé, P. (1946) *Les Réserves et la Régulation de L'Avenir* Hermann and Cle, Paris.

[56] McFadden, D. (1989) "A Method of Simulated Moments for Estimation of Discrete Response Models without Numerical Integration" *Econometrica* **57** 995–1026.

[57] Nemirovsky, A. S. and D.B. Yudin (1983) *Problem Complexity and Method Efficiency in Optimization* Wiley, New York.

[58] Nielsen, T. D. and J. Jaffray (2004) "Dynamic Decision Making without Expected Utility: An Operational Approach" manuscript, Université Paris 6.

[59] Pakes, A. (1986) "Patents as Options: Some Estimates of the Values of Holding European Patent Stocks" *Econometrica* **54** 755–784.

[60] Pakes, A. and P. McGuire (1994) "Computing Markov Perfect Nash Equilibrium: Numerical Implications of a Dynamic Differentiated Product Model" *RAND Journal of Economics* **25** 555–589.

[61] Pakes, A. (2001) "Stochastic Algorithms, Symmetric Markov Perfect Equilibria and the 'Curse' of Dimensionality" *Econometrica* **69** 1261–1281.

[62] Penrose, R. (1989) *The Emperor's New Mind* Penguin Books, New York.

[63] Pesendorfer, M. and P. Schmidt-Dengler (2003) "Identification and Estimation of Dynamic Games" manuscript, University College London.

[64] Phelan, C. (2006) "Government Trust and Betrayal" *Journal of Economic Theory* forthcoming.

[65] Pollak, R. A. (1968) "Consistent Planning" *Review of Economic Studies* **35** 201–208.

[66] Pollard, D. (1989) "Asymptotics via Empirical Processes" *Statistical Science* **4** 341–386.

[67] Puterman, M.L. (1994) *Markovian Decision Problems* John Wiley and Sons, New York.

[68] Rust, J. (1985) "Stationary Equilibrium in a Market for Durable Goods" *Econometrica* **53** 783–805.

[69] Rust, J. (1987) "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher" *Econometrica* **55** 999-1033.

[70] Rust, J. (1988) "Maximum Likelihood Estimation of Discrete Control Processes" *SIAM Journal on Control and Optimization* **26** 1006-1024.

[71] Rust, J. (1994) "Structural Estimation of Markov Decision Processes" in R.F. Engle and D.L. McFadden (eds.) *Handbook of Econometrics Volume IV* Amsterdam, Elsevier.

[72] Rust, J. (1996) "Numerical Dynamic Programming in Economics" (1996) in H. Amman, D. Kendrick and J. Rust (eds.) *Handbook of Computational Economics* Elsevier, North Holland.

[73] Rust, J. (1997) "Using Randomization to Break the Curse of Dimensionality" *Econometrica* **65** 487–516.

[74] Rust, J. and C. Phelan (1997) "How Social Security and Medicare Affect Retirement Behavior in a World with Incomplete Markets" (1997) *Econometrica* 65, 781–832.

[75] Rust, J. (2002) “Is There a Curse of Dimensionality for Contraction Fixed Points in the Worst Case?” (2002) (with J.F. Traub and H. Woźniakowski) *Econometrica* **70** 285–329.

[76] Rust, J. and M. Santos (2004) “Convergence Properties of Policy Iteration” *SIAM Journal on Control and Optimization*.

[77] Rust, J. and G. J. Hall (2005) “The (S,s) Rule is an Optimal Trading Strategy in a Class of Commodity Price Speculation Problems” *Economic Theory*

[78] Ruud, P. (2006) “Simulation Based Inference” *New Palgrave Dictionary of Economics*.

[79] Sargent, T.J. (1978) “Estimation of Dynamic Labor Demand Schedules under Rational Expectations” *Journal of Political Economy* **86** 1009-1044.

[80] Sargent, T.J. (1981) “Interpreting Economic Time Series *Journal of Political Economy* **89** 213–248.

[81] Sargent, T.J. (1987) *Dynamic Macroeconomic Theory* Harvard University Press, Cambridge, Massachusetts.

[82] Selten, R. (1975) “Reexamination of the Perfectness Concept for Equilibrium Points in Extensive Games” *International Journal of Game Theory* **4** 25–55.

[83] Skiadis, C. (1998) “Recursive Utility and Preferences for Information” *Economic Theory* **12** 293–312.

[84] Stokey, N.L. and R. E. Lucas Jr., with E.C. Prescott (1989) *Recursive Methods in Economic Dynamics* Harvard University Press, Cambridge, Massachusetts.

[85] Streufert, P. (1998) “Recursive Utility and Dynamic Programming” in S. Barbera, P. Hammond, and C. Seidl (editors) *Handbook of Utility Theory, Volume 1* chapter 3 93–121.

[86] Strotz, R. H. (1956) “Myopia and Inconsistency in Dynamic Utility Maximization” *Review of Economic Studies* **23** 165–180.

[87] Tauchen, G. and R. Hussey (1991) “Quadrature-based Methods for Obtaining Approximate Solutions to Nonlinear Asset Pricing Models” *Econometrica* **59** 371–396.

[88] Todd, P, and K.I. Wolpin (2005) “Ex Ante Evaluation of Social Programs” manuscript, University of Pennsylvania.

[89] Traub, J.F. and Werschulz (1998) *Complexity and Information* Cambridge University Press, Cambridge, UK.

[90] von Neumann, J. and O. Morgenstern (1944) *Theory of Games and Economic Behavior* 3rd edition, Princeton University Press, Princeton, N.J.

[91] Wald, A. (1947) “Foundations of a General Theory of Sequential Decision Functions” *Econometrica* **15** 279–313.

[92] Wald, A. (1947) *Sequential Analysis* Dover Publications, Inc. New York.

[93] Wald, A. and J. Wolfowitz (1948) "Optimum Character of the Sequential Probability Ratio Test" *Annals of Mathematical Statistics* **19** 326–339.

[94] Wolpin, K. (1984) "An Estimable Dynamic Stochastic Model of Fertility and Child Mortality" *Journal of Political Economy* **92-5** 852–874.