Numerical Dynamic Programming

entry for consideration by the

Wiley StatsRef: Statistics Reference Online

John Rust, *Georgetown University*[*]

June 22, 2016

[*]Department of Economics, Georgetown University, Intercultural Center, Washington, DC 20057-1036, phone: (202) 687-6806, e-mail: jr1393@georgetown.edu.

Dynamic Programming (DP) is a name coined by Bellman (1957) for the recursive method of backward induction for computing optimal decision rules to *sequential decision problems* (SDP) (which Bellman called "multi stage decision processes" StatsRef 022232) and *subgame perfect equilibria* of dynamic multi-agent games and competitive equilibria of dynamic economic models. A *decision rule* (or *strategy* in a game) is a function that specifies an action (or probability distribution over actions in a game) at each stage or state of the decision process (or game). A huge class of problems can be formulated as SDPs or dynamic games, and can accommodate *risk and uncertainty* (under the criterion to maximize expected payoffs) and *learning* (e.g. Bayesian updating) when aspects of the environment or law of motion of the problem are not completely known by the decision maker (DM). Though the term "dynamic" is appropriate since DP is typically used to solve intertemporal decision problems in discrete time (where the index for backward induction is t, the time variable) it has been extended to problems in continuous time and problems with no obvious intertemporal structure (e.g. Djikstra's algorithm for the shortest path problem) if there is "directionality" that can be encoded as directed graph over the states of the SDP or game.

DP is a *decomposition principle* that breaks the solution of a complex overall problem into a recursively linked set of simpler subproblems. This is captured in Bellman's *Principle of Optimality* "An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regards to the state resulting from the first decision." (Bellman, 1957, p. 56). It is intuitive that solving a SDP or dynamic game by backward induction results in an optimal decision rule (or subgame perfect equilibrium) that satisfies Bellman's Principle of Optimality (or in dynamic games, the "one shot deviation principle"). The Principle of Optimality appears to depend on time t as the index for backward induction, but Iskhakov, Schjerning and Rust (2016) introduced *state recursion,* a generalization of backward induction for SDPs and dynamic games. They defined a partial ordering over states of the problem that formalizes the intuitive concept of *directionality.* This leads to a decomposition of the SDP into *stages* (indexed by τ) that provides an alternative to the usual time index t as the index for the backward induction process. Though optimal decision rules to SDPs can be formally recast and solved as static, "open loop" optimization problems, it is typically much more effective to solve them using DP (or state recursion when applicable) because it decomposes the solution to the overall problem into the recursive solution of a sequence of smaller, more tractable subproblems.[1]

DP arose out of early work on statistical decision theory and game theory in the 1940s when the first digital computers were developed (Neumann and Morgenstern 1944, Rust, 2008, Dyson 2012, and StatsRef 01909 and 00215). Though the term "programming" is suggestive of computer programming, the early work on DP was mostly theoretical, as the early computers were not sufficiently powerful to solve most practical DP problems. However the usefulness and flexibility of DP for formulating and solving a wide class of problems and games of practical importance and relevance was quickly recognized in fields such as operations research (inventory control, and optimal investment and replacement or machinery, and resource management, see e.g. StatsRef 04185, 00079, 07704, 04714), and engineering (management of electricity grids and power generation, communications and network routing, and other examples discussed in Powell 2010). There has been extensive theoretical and empirical applications of SDPs and dynamic games in economics, where firms and consumers are assumed to be rational agents whose behav-

[1] Iskhakov, Rust and Schjerning (2016) developed a related "recursive lexicographical search" (RLS) algorithm that can compute and enumerate all possible equilibria for the subclass of "directional dynamic games" (DDGs) that generalizes and exploits the benefits of the DP approachx, by decomposing the complex problem of finding all equilibria to an overall dynamic game into the easier problem of finding all equilibria of its recursively linked subgames. Chess is a classic example of a directional game: the number of pieces on the board never increases as the game unfolds. The RLS algorithm has been used to enumerate hundreds of millions of equilibria even in relatively simple dynamic games with small state spaces.

ior are governed by optimal (or equilibrium) decision rules (Stokey and Lucas 1989, Rust 1996). This, in turn, as lead to an empirical literature on numerical solution and estimation of "dynamic structural models" in econometrics due to their value in making causal inferences and counterfactual policy predictions (Adda and Cooper, 2003, Rust 1994, 2014).

The huge number of applications of SDPs induced a demand for more powerful numerical methods to solve or approximate the solutions to large scale DP problems on digital computers, which are themselves improving at exponential rates due to the digital revolution and numerous technological improvements embodied in "Moore's Law." However the size or dimensionality of many practical DP problems can be enormous. For example, there has been extensive work on optimal inventory policy dating to the original work by Arrow, Harris and Marshak (1951). Scarf (1960) proved that the optimal policy is an (S,s) rule and Rust and Hall (2005) showed that a generalized version of the (S,s) rule is optimal when the price at which retailers acquire inventory from producers or wholesalers is time-varying. Then the (S,s) thresholds are functions of the wholesale price p, and for a retailer selling just one type of product, finding these functions involves solving a $d = 2$ dimensional DP problem, with states p (wholesale price) and q (current inventory). However the Food Marketing Institute reports that in 2014 a typical US grocery store stocked over 42,000 individual items. Thus, the DP problem for a grocery store that allows for correlated wholesale prices and shelf space contraints (so the problem cannot be decomposed into 42,000 separate two-dimensional DP subproblems), has more than $d = 84,000$ dimensions!

Bayesian learning is used In DP problems where the DM does not observe the true state or does not completely know the law of motion governing the state variables, so the posterior distribution over these uncertain quantities is one of the state variables of the DP problem. Unless the problem satisfies restrictive conditions for the posterior to be a member of a conjugate family (i.e. a parametric family such as Gaussian distributions), it constitutes an *infinite dimensional state variable.* Even if we make a finite approximation to this posterior (e.g. a multinomial distribution), the result is a very high finite dimensional DP problem. Even in problems or games where the state space is finite, the number of possible values can be huge. The number of possible board positions in Chess is about 10^{50} and there are more than 10^{100} positions in the game of Go — more than the number of atoms in the observable universe according to *Wikipedia*.

Bellman (1957) coined the term *curse of dimensionality* to describe the exponential increase in computer time to solve increasingly realistic and detailed DP problems. Bellman did not establish whether the curse of dimensionality constitutes a fundamental "limit to science" or is just a temporary obstacle that could be circumvented by better solution algorithms and faster computers.[2] A subsequent literature on *information-based complexity* in computer science (see StatsRef 02992, 00408 and Traub and Werschulz, 1998) formally defined the "curse of dimensionality" using an explicit model of computation that recognizes that some mathematical operations (e.g. integration) require an infinite amount of information on the underlying function to obtain an exact answer, whereas actual computers and algorithms can only use a finite amount of information to find an approximate solution. Using this framework, computer scientists succeeded in characterizing many mathematical problems that are subject to the curse of dimensionality — i.e. they proved it constitutes a fundamental "limit to science" that cannot be circumvented by *any algorithm* regardless of how clever it might be.

Computer scientists use the term *intractable* to describe any mathematical problem that is subject to the curse of dimensionality: these are problems where the worst case computational cost of finding an ε-approximation to the true solution increases exponentially fast in d, the number of continuous variables or

[2]Bellman (1957) seemed to be optimistic that the curse of dimensionality could be overcome: "Assume, however, that we have circumvented all these difficulties and have attained a certain computational nirvana." (p. ix).

dimensionality of the problem.[3] Mathematical problems whose complexity do not increase exponentially fast in d (such as at rate $(1/\varepsilon)^K$ where K is independent of d) are said to be *tractable.* Nemirovsky and Yudin (1983) proved that ordinary mathematical programming problems (i.e. optimization of smooth real-valued functions of d variables) are subject to a curse of dimensionality, and this curse cannot even be broken using stochastic algorithms (e.g. stochastic search algoirthms such as simulated annealing). Since ordinary mathematical programming problems are a subclass of the class of all SDP problems, it follows that there is an unvoidable curse of dimensionality for solving DP problems as well. Chow and Tsitsiklis (1989) derived a complexity bound that shows that complexity increased exponentially in d_s (dimension of the continuous state variables) and d_c (dimension of the continuous decision variables, c).

However there are subclasses of SDP problems that have *special structure* that can be exploited to break the curse of dimensionality. For example the class of linear-quadratic-Gaussian (LQG) SDPs (where the payoff function is quadratic and the law of motion of linear stochastic difference equations with additive Gaussian shocks) can be solved exactly in polynomial time (i.e. where solution time is a polynomial function of the number of state and control variables).[4] Gittins (1979) showed that the optimal search strategy for the *multi-armed bandit problem* (statsRef 00297) takes the form of a *reservation price rule* which can be computed by decomposing the overall problem into separate much simpler individual optimal stopping problems. Rust (1997) proved that the curse of dimensionality can be broken using a *random multigrid algorithm* for the class of *dynamic discrete choice models* (i.e. SDPs where the decision variable is restricted to a finite set of alternatives). He showed that the upper bound on the computer time to provide a stochastic uniform ε-approximation is proportional to $(1/\varepsilon)^4$ independent of the dimension d, whereas the Chow-Tsitsiklis (1989) *lower* bound on complexity is proportional to $(1/\varepsilon)^{2d}$ when only deterministic algorithms are allowed. The intuition for this result is there is no longer an unavoidable curse of dimensionality associated with the "optimization subproblem" of a DP problem when the DM faces only a finite number of possible choices in each state. There is still a potential curse of dimensionality associated with multivariate integration and approximation of the value function and decision rules, but Rust used empirical process methods (statsRef 02986) to show how that Monte Carlo integration can be used to provide uniform approximations that break the curse of dimensionality associated with these latter subproblems similar to the way Monte Carlo integration breaks the curse of dimensionality of approximating a single high dimensional integral.

The disadvantage of using randomized algorithms such as Monte Carlo integration is the stochastic noise in the solution, although this noise can be made as small as desired by using a larger number of random draws that form the "stochastic grids" in the multigrid method. Quasi monte carlo methods based on "low discrepancy sequences" can ameliorate the undesirable effects of randomization and result in faster rates of convergence, while still succeeding in breaking the curse of dimensionality in certain subclasses of problems (Rust, Traub and Woźniakowsky, 2002). However the state of the art for providing relatively precise approximate solutions to DP problems of moderate dimension is to use non-stochastic methods including multivariate quadrature (for numerical integrations) and various types of fast interpolation and sparse adaptive grids such as the Smolyak algorithm (Brumm and Scheidegger, 2016).

The literature on numerical methods for DP problems that are actually *used in practice* is too vast to do justice to it in a short survey, so what follows is just a brief overview of the best known methods

[3]Formally, a mathematical problem is intractable if there is lower bound on the complexity function (the minimal cost of finding a ε-approximation to the true solution in the worst case using *any* algorithm that uses only a finite amount of information about the particular problem being solved, that increases as $(1/\varepsilon)^d$ where d is the dimensionality of the problem.

[4]In addition, problems where the state variable is observed with noise (where the Kalman filter is required, statsRef 04585) can also be solved in polynomial time.

with some discussion of promising new methods. Much of the work has been done in the context of Markovian Decision Problems (MDPs), which are SDPs where the law of motion follows a (controlled) Markov process and the DM maximizes an expected discounted stream of current and future payoffs. In finite horizon problems DP amounts to the usual backward induction process starting at the last period and working backward in time. In problems that are stationary and infinite horizon the solution to the MDP is equivalent to solving the *Bellman equation* which is a functional equation for the *value function* $V(s)$ that provides the expected discounted payoffs under an optimal policy conditional on being in state s. The *decision rule* $\delta(s)$ is the action that maximizes the expected payoff to the DM in state s. Solving the MDP amounts to finding an suitable ε approximation to the value function V and the decision rule δ, though these functions are "dual" to each other in the sense that V can be recovered from δ and vice versa.

It has been known since the work of Blackwell (1965) and DeNardo (1967) that there is a unique solution V to Bellman's equation in bounded problems satisfying mild regularity conditions. We can write the Bellman equation compactly as $V = \Gamma(V)$ where Γ is a *contraction mapping* known as the *Bellman operator* Γ defined by

$$\Gamma(V)(s) = \max_{d \in D(s)} [u(s,d) + \beta \int V(s')p(s'|s,d)], \tag{1}$$

where $\beta \in (0,1)$ is the discount factor, $D(s)$ is the set of feasible choices for the DM in state s, $u(s,d)$ is the payoff (utility) function, and $p(s'|s,d)$ is a Markov transition probablity kernel providing the law of motion for the controlled Markov process.

Since Γ is a contraction mapping, its fixed point V can be approximated arbitrarily closely from any starting point via the method of *successive approximations* which converges to the true solution in the limit as the number of iterations tends to infinity. An approximate solution via successive approximations (starting from an initial guess of $V = 0$) is equivalent to solving an approximate finite horizon truncation to the infinite horizon problem by backward induction. With bounded payoffs and discounting, the error from any such truncation can be made as small as desired. Howard (1960) introduced the *policy iteration algorithm* that computes V and δ *exactly* (up to machine precision) when the state and action spaces are finite. Puterman (1994) showed that policy iteration is equivalent to solving Bellman's equation via Newton's method, i.e. as a zero of the nonlinear operator $F(V) = 0$ where $F(V) = V - \Gamma(V)$. Bertsekas (2005, 2012) and Puterman (1994) describe other iterative methods for approximating V (e.g. linear programming, Gauss-Siedel, etc).

The numerical problems of approximating V and δ are much more complicated when the problem has continuous state variables and/or continuous decisions. When there infinitely many states the value function becomes an infinite dimensional object (i.e. an element of a function space, such as the Banach space $C(S)$ of bounded, continuous functions on the state space S when S, u and p satisfy appropriate regularity conditions such as S is a compact subset of R^k, and u and p satisfy continuity conditions). Then V and δ can only be approximated based on information from a finite number of points $\{s_1, \ldots, s_N\}$ in the state space. Given estimates of V and δ at these N points, their values at other values of $s \in S$ must be either *interpolated* or *approximated.* There are a huge number of ways to do this: Judd (1998) is a good reference to many of the commonly used methods. For example, a common approach is via *parametric approximations* to V, such as expressing V as a linear combination of polynomials in s such as *Chebyshev polynomials* with coefficients θ to be determined, e.g.

$$V_\theta(s) = \sum_{j=1}^{J} \theta_j \rho_j(s) \tag{2}$$

where ρ_j is the j^{th} Chebyshev polynomial (statsRef 02221). However many other "approximation architectures" can also be considered including nonlinear (in θ) approximations such as local polynomials (statsRef 02716), splines (statsRef 07567), and neural networks (statsRef 09037). Given a choice of interpolation or approximation architecture, there are different ways to find the θ coefficients to best approximate the true solution V. For example we can use *nonlinear least squares* (statsRef 03220) and approximate V with $V_{\hat{\theta}}$ where $\hat{\theta}$ is given by

$$\hat{\theta} = \underset{\theta}{argmin} \|V_\theta - \hat{\Gamma}(V_\theta)\|. \tag{3}$$

There are additional choices about how implement an "approximate Bellman operator" $\hat{\Gamma}$ that involve a variety of choices of numerical integration methods (to approximate the expectation operator in equation (1)) and numerical optimization methods (to perform the constrained maximization over d in (1)) in problems with continuous control variables. We also need to approximate the norm $\|\cdot\|$ as the theory typically uses the supremum norm but this can only be approximated when the state space S is infinite.

Iterative techniques similar to successive approximations are also used, such as forming a sequence of estimates $\{V_{\theta_j}\}$ via the recursion $\theta_j = argmin_\theta \|V_\theta - \hat{\Gamma}(V_{\theta_{j-1}})\|$, which amounts to a type of successive approximation in terms of the $\{\theta_j\}$ coefficients. One reason to prefer an iterative approach to (3) is speed: under the L_2 norm, θ_j can be calculated by ordinary least squares. However it takes considerable work to to prove that various intuitively plausible numerical procedures actually work in the sense that for a suitable sequence of grid points $\{s_1, \ldots, s_N\}$ and sequence of parametrizations of V in terms of J parameters as V_θ where $\theta \in R^J$, that $\lim_{N,J\to\infty} \|V_\theta - V\| = 0$, i.e that the numerical method is *consistent* in analogy to how this term is used in non-parametric estimation. It is even harder to establish computable error bounds similar to ones that exist for finite state MDPs (McQueen, 1966, Porteus, 1975), so there is a need for theoretical analysis of the consistency and convergence rates of the myriad of methods used in practice.

It is still more difficult to establish consistency of numerical procedures for MDPs involving unbounded state spaces and unobounded payoff functions, problems that are often encountered in economic applications. Since any numerical solution will generally only involve evaluations of V and δ a finite number of points of S, a numerical solution will necessarily be forced to rely on extrapolations that may not be valid and result in erroneous approximations for V and δ outside the compact subset of the state space where the grid points used to solve the problem are located. In addition considerable care must be taken to employ suitable *regularization methods* (StatsRef 07415) to avoid the equivalent problem of "overfitting" in non-parametric statistics. For example using tensor products of high degree polynomials to approximate V can produce oscillations that may result in a good fit (such as if a nonlinear least squares criterion such as (3) is employed), but very poor approximations to the true decision rule δ and V.

There are close connections between the literatures on optimal control, DP, dynamic games, and artificial intelligence (AI) (including its modern incarnation, machine learning StatsRef 00507, 05023) as well as an older literature on "cybernetics" and modern work on robotics. The main difference between the former literatures and AI is that the former focuses on finding *optimal solutions* to these problems (or equilibrium solutions in the cases of dynamic games), whereas the AI literature is content to find *intelligent, adaptive decision rules* even if they are not necessarily optimal or equilibrium solutions. Simon (1956) referred to these decision rules as *satisficing* and viewed human behavior through this lens on the grounds that in sufficiently complex environments it is difficult or impossible to determine what an optimal decision rule might be.[5] Thus, the AI literature has side-stepped formally justifiable procedures for finding optimal/equilibrium solutions in favor of informal procedures such as *heuristics* and "rules of thumb" that are "good enough" even though potentially suboptimal.

[5] "Evidently, organisms adapt well enough to satisfice; they do not, in general, optimize." p. 129.

There are particularly close connections between a branch of the AI literature on *reinforcement learning* which is also known as *approximate dynamic programming* (Sutton and Barto, 1996, Powell, 2010, Bertsekas, 2011). In this literature, "learning" means approximating the optimal decision rule to a MDP problem via iterative, stochastic algorithms. Examples of reinforcement learning algorithms include *Q-learning* (Watkins, 1989) and *real time dynamic programming* (Barto, Bradtke and Singh, 1995). These algorithms are also described as *model free* since they do not depend on knowledge of the law of motion for the stochastic states of the problem (e.g. the transition probability $p(s'|s,d)$ of the MDP, in equation (1) provided it is possible to *simulate realizations* from this transition probability. These algorithms also update the decision rule *in real time,* while the DM is actually making decisions.

Q-learning takes its name from the function $Q(s,d)$, equal to the right hand side of the Bellman equation (1),

$$Q(s,d) \equiv u(s,d) + \beta \int V(s')p(s'|s,d). \tag{4}$$

If we knew the function Q, then the optimal decision in any state $s \in S$ is simply the alternative $d \in D(s)$ that yields the highest value of $Q(s,d)$, i.e. $\delta(s) = argmax_{d \in D(s)} Q(s,d)$. Consider a DM who is in state s_t at time t and who must decide which decision to take while also trying to improve their performance via "learning by doing." The DM takes action $d_t = argmax_{d \in D(s_t)} Q_t(s_t,d)$ and observes a realization of the next period state, a draw $\tilde{s}$ from the transition probability $p(s'|s_t,d_t)$. Q-learning results in a sequence of Q functions given by

$$Q_{t+1}(s_t,d_t) = (1-\gamma_t)Q_t(s_t,d_t) + \gamma_t[u(s_t,d_t) + \beta \max_{d \in D(\tilde{s})} Q_t(\tilde{s},d)], \tag{5}$$

where $\gamma_t \geq 0$ is a stepsize sequence. Note that the values of Q_t for all other pairs $(s,d) \neq (s_t,d_t)$ remain unchanged. Q-learning can be viewed as a type of asynchronous stochastic approximation, whose almost sure convergence can be be established by martingale limit theorems under standard conditions on the stepsize (decreasing to zero, but not too fast) and under the assumption that during the learning or "training" process, all pairs (s,d) are visited infinitely often (Tsitsiklis, 1994).

These types of convergence results provide some confidence that some of the intuitively conjectured learning algorithms from the AI/reinforcement learning literature can actually converge to optimal solutions. The earliest consistency results required finite state and action spaces, and rates of convergence to the optimal solution can be slow, so substantial "training" is required before acceptable performance can be expected from these algorithms (i.e. before Q_t is close to the true Q). This raises questions as to whether Q-learning and other reinforcement learning algorithms can be extended to problems with large state and action spaces, such as problems with continuous state and decision variables. In these high dimensional problems, the general approach is to use parametric approximations of the Q function similar to approaches we described above for approximating the value function V. Call the parameterized Q function Q_θ. The goal is to find flexible parametric approximations that depend on a low dimensional parameter vector $\theta \in R^n$ where n is relatively small. Simulations of the controlled process result in "data" that can be used to fit the θ parameters of $Q_\theta(s,d)$, resulting in an iterative, stochastic sequence of $\{\theta_t\}$ coefficients that is similar to the sequence of "successive approximations" of the θ coefficients in the value function paramterization V_θ described above. For example, one commonly used iterative approach for producing a $\{\theta_t\}$ sequence is

$$\theta_{t+1} = \theta_t + \gamma_t \nabla Q_{\theta_t}(s_t,d_t)\left(u(s_t,d_t) + \beta \max_{d' \in D(\tilde{s})} Q_{\theta_t}(\tilde{s},d') - Q_{\theta_t}(s_t,d_t)\right), \tag{6}$$

where $\nabla Q_\theta(s,d)$ is the gradient of $Q_\theta(s,d)$ with respect to θ, and as in the finite state version of Q-learning above, $d_t = argmax_{d\in Q_{\theta_t}}(s_t,d)$ and $\tilde{s}$ is a realization of the transition probability $p(s'|s_t,d_t)$ and $\gamma_t \geq 0$ is a stepsize parameter.

Neural networks are a commonly used approximation architecture since they provide a flexible parametric class of functions that can approximate a wide class of functions using relatively few parameters θ. For example Barron (1994) proved that single layer neural networks with n nodes can provide approximations to classes of functions with integrated squared error of $O(C/n)$, where C is a measure of the smoothness of the function being approximated. Further, with a "training dataset" of N observations, he showed that the mean squared approximation error for the optimally chosen number of nodes is $O(C\sqrt{(d/N)log(N)})$. Thus, to first order the error decreases at rate $1/\sqrt{N}$, and this means the neural network is able to break the usual curse of dimensionality of non-parametric regression for the class of functions Barron considered, since the non-parametric bounds of Stone (1982) for nonparametric estimation of functions with s derivatives is of order $(1/N)^{2s/(2s+d)}$, i.e. the rate of convergence decreases monotonically in the dimension d.

These attractive properties of neural networks spurred a productive line of research in the reinforcement learning and approximate DP literatures known as *neuro-dynamic programming* (Bertsekas and Tsitsiklis 1996). There are a huge number of specific methods that fall within this rubric, though not much is known about the consistency and rates of convergence of many of the popular learning algorithms, including Q learning with neural networks. However there has been a great deal of practical experimentation with these methods with some considerable recent success. For example teams at Google have coupled Q-learning with *deep neural networks* — multilayer neural networks "in which several layers of nodes are used to build up progressively more abstract representations of the data, have made it possible for artificial neural networks to learn concepts such as object categories directly from raw sensory data" (Mnih *et. al.* 2015, p. 529). They tested these "deep Q-networks" (DQN) "on the challenging domain of classic Atari 2600 games. We demonstrate that the deep Q-network agent, receiving only the pixels and the game score as inputs, was able to surpass the performance of all previous algorithms and achieve a level comparable to that of a professional human games tester across a set of 49 games" (p. 529).

Even more impressively, another team at Google trained an algorithm, *AlphaGo,* to play the game of Go using DQN coupled with tree search "that plays at the level of the strongest human players, thereby achieving one of artificial intelligence's 'grand challenges"'(p. 488). In fact, in March 2016 *AlphaGo* made headlines by defeating the World champion human Go player, Le Se-dol, winning 4 games in a 5 game match. The algorithm was initially trained on a database of 30 million board positions from human games and then further "self-trained" by playing millions of additional games against other instances of itself. Since *AlphaGo* employs fairly deep 13 layer neural networks, extensive training was required and "Evaluating policy and value networks requires several orders of magnitude more computation than traditional search heuristics." (p. 487).

These types of encouraging examples suggest that neuro dynamic programming could be a very promising approach to the approximate solution of many other high dimensional DP problems. "With a good deal of justification, it claims to deal effectively with the dual curses of dynamic programming and stochastic optimal control: Bellman's *curse of dimensionality* (the exponential computational explosion with the problem dimension is averted through the use of parametric computational representations of the [value] function), and the *curse of modeling* (an explicit system model is not needed and a simulator can be used instead)." (Bertsekas and Tsitsiklis, 1996, p. xi). However there is no formal proof of this and indeed, the complexity bounds of Nemirovsky and Yudin (1983) and Chow and Tsitsiklis (1989) imply that *no algorithm* (including any neurodynamic programming algorithm) can succeed in breaking the curse

of dimensionality for the general class of MDPs with continuous state and control variables.

So the current state of the art seems to be this: there are a number of good, reliable, deterministic methods that provide relatively accurate solutions to low to moderate dimensional problems. However for very complicated high dimensional problems that lack any exploitable "special structure," we must be prepared for the possibility that there will be a fair amount of innaccuracy and stochastic noise in an approximate solution and it will be difficult to assess how far it is from the optimal solution. The moral might be to try to recognize and exploit the special structure of a problem whenever possible, but failing that, if you are attempting to solve very large scale problems, be ready to make pragmatic and sometimes *ad hoc* shortcuts and simplifications, adopt reasonable heuristics, and be prepared to admit that "the best is the enemy of the good."

References

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

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

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

[4] Barron, A. (1994) "Approximation and Estimation Bounds for Neural Networks" *Machine Learning* **14** 115–133.

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

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

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

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

[9] Bertsekas, D. P. (2005) *Dynamic Programming and Optimal Control, volume 1* Athena Scientific, Belmont, Massachusetts.

[10] Bertsekas, D.P. (2011) "Approximate policy iteration: a survey and some new methods" *Journal of Control Theory and Applications* **9-3** 310–335.

[11] Bertsekas, D. P. (2012) *Dynamic Programming and Optimal Control, volume 2, Approximate Dynamic Programming* Athena Scientific, Belmont, Massachusetts.

[12] Bertsekas, D. P. (2013) *Abstract Dynamic Programming* Athena Scientific, Belmont, Massachusetts.

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

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

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

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

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

[18] Brumm, J. and S. Scheidegger (2016) "Using Adaptive Sparse Grids to Solve High Dimensional Dynamic Models" manuscript, University of Zurich.

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

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

[21] Dyson, G. (2012) *Turing's Cathedral* Pantheon Books, New York.

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

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

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

[25] Iskhakov, F. J. Rust and B. Schjerning (2016) "Recursive Lexicographical Search: Finding All Markov Perfect Equilibria of Finite State Directional Dynamic Games" *Review of Economic Studies* **83-2** 658–703.

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

[27] 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.

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

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

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

[31] 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.

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

[33] Mnih, V. *et. al.* (2015) "Human-level control through deep reinforcement learning" *Nature* **518** 529–532.

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

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

[36] Powell, W.B. (2010) *Approximate Dynamic Programming Solving the Curses of Dimensionaity* John Wiley & Sons, Hoboken, New Jersey.

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

[38] 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.

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

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

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

[42] 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*

[43] Rust, J. (2008) "Dynamic Programming" *New Palgrave Dictionary of Economics* Second Edition, Palgrave MacMillan, London.

[44] Rust, J. (2015) "The Limits to Inference *with* Theory: A Review of Wolpin (2013)" *Journal of Economic Literature* **52-3** 820–850.

[45] Scarf, H. (1960) "The optimality of (S,s) policies in the dynamic inventory problem" in K. Arrow, P. Suppes and S. Karlin (eds.) *Mathematical Methods in the Social Sciences* Stanford University Press, Stanford.

[46] Silver, D. *et. al.* (2016) "Mastering the game of Go with deep neural networks and tree search" *Nature* **529** 485–489.

[47] Simon, H.A. (1956) "Rational Choice and the Structure of the Environment" *Psychological Review* **63-2** 129–138.

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

[49] Stone, C. (1982) "Optimal global rates of convergence for nonparametric estimators" *Annals of Statistics* **10** 1040–1053.

[50] Sutton, R.S. and Barto, A. G. (1998) *Reinforcement Learning: An Introduction* MIT Press, Cambridge, Massachusetts.

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

[52] Tsitsiklis, J. (1994) "Asynchronous Stochastic Approximation and Q-Learning" *Machine Learning* **16** 185–202.

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

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

[55] Watkins, C. (1989) “Learning from Delayed Rewards” Ph.D. thesis, University of Cambridge.

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