European Journal of Control (2005)11:310–334

European Journal of Control

Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC*

Dimitri P. Bertsekas**

Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, USA

We survey some recent research directions within the field of approximate dynamic programming, with a particular emphasis on rollout algorithms and model predictive control (MPC). We argue that while they are motivated by different concerns, these two methodologies are closely connected, and the mathematical essence of their desirable properties (cost improvement and stability, respectively) is couched on the central dynamic programming idea of policy iteration. In particular, among other things, we show that the most common MPC schemes can be viewed as rollout algorithms and are related to policy iteration methods. Furthermore, we embed rollout and MPC within a new unifying suboptimal control framework, based on a concept of restricted or constrained structure policies, which contains these schemes as special cases.

Keywords: dynamic programming, stochastic optimal control, model predictive control, rollout algorithm

1. Introduction

We consider a basic stochastic optimal control problem, which is amenable to a dynamic programming solution, and is considered in many sources (including the author's dynamic programming textbook [14], whose notation we adopt). We have a discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, \tag{1.1}$$

where x_k is the state taking values in some set, u_k is the control to be selected from a finite set $U_k(x_k)$, w_k is a random disturbance and f_k is a given function. We assume that each w_k is selected according to a probability distribution that may depend on x_k and u_k, but not on previous disturbances. The cost incurred at the k-th time period is denoted by $g_k(x_k, u_k, w_k)$. For mathematical rigor (to avoid measurability questions), we assume that w_k takes values in a countable set, but the following discussion applies qualitatively to more general cases.

We initially assume perfect state information, i.e. that the control u_k is chosen with knowledge of the current state x_k; we later discuss the case of imperfect state information, where u_k is chosen with knowledge of a history of measurements related to the state. We, thus, initially consider policies $\pi = \{\mu_0, \mu_1, \ldots\}$, which at time k map a state x_k to a control $\mu_k(x_k) \in U_k(x_k)$. We focus primarily on an N-stage horizon problem where k takes the values $0, 1, \ldots, N-1$, and there is a terminal cost $g_N(x_N)$ that depends on the terminal state x_N. The cost-to-go of π starting from a state x_k at time k is denoted by

$$J_k^\pi(x_k) = E\left\{ g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, \mu_i(x_i), w_i) \right\}. \tag{1.2}$$

The optimal cost-to-go starting from a state x_k at time k is

$$J_k(x_k) = \inf_\pi J_k^\pi(x_k),$$

*Many thanks are due to Janey Yu for helpful comments. Research supported by NSF Grant ECS-0218328.
**E-mail: dimitrib@mit.edu

Received 15 June 2005; Accepted 30 June 2005.
Recommended by E.F. Camacho, R. Tempo, S. Yurkovich, P.J. Fleming

and it is assumed that $J_k^{\pi}(x_k)$ and $J_k(x_k)$ are finite for all x_k, π and k. The cost-to-go functions J_k satisfy the following recursion of dynamic programming (DP)

$$J_k(x_k) = \inf_{u_k \in U_k(x_k)} E\{g_k(x_k, u_k, w_k) + J_{k+1} \times (f_k(x_k, u_k, w_k))\}, \quad k = 0, 1, \ldots, N-1, \tag{1.3}$$

with the initial condition

$$J_N(x_N) = g_N(x_N).$$

Our discussion applies with minor modifications to infinite horizon problems, with the DP algorithm replaced by its asymptotic form (Bellman's equation). For example, for a stationary discounted cost problem, the analog of the DP algorithm (1.3) is

$$J(x) = \inf_{u \in U(x)} E\{g(x, u, w) + \alpha J(f(x, u, w))\}, \quad \forall x, \tag{1.4}$$

where $J(x)$ is the optimal (α-discounted) cost-to-go starting from x.

An optimal policy may be obtained in principle by minimization in the right-hand side of the DP algorithm (1.3), but this requires the calculation of the optimal cost-to-go functions J_k, which for many problems is prohibitively time-consuming. This has motivated approximations that require a more tractable computation, but yield a suboptimal policy. There is a long history of such suboptimal control methods, and the purpose of this paper is to survey some of them, to discuss their connections, and to place them under the umbrella of a unified methodology. Although we initially assume a stochastic model with perfect state information, much of the subsequent material is focused on other types of models, including deterministic models.

A broad class of suboptimal control methods, which we refer to as *approximate dynamic programming* (ADP), is based on replacing the cost-to-go function J_{k+1} in the right-hand side of the DP algorithm (1.3) by an approximation $\tilde{J}_{k+1}$, with $\tilde{J}_N = g_N$. Thus, this method applies at time k and state x_k a control $\overline{\mu}_k(x_k)$ that minimizes over $u_k \in U_k(x_k)$

$$E\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k))\}.$$

The corresponding suboptimal policy $\overline{\pi} = \{\overline{\mu}_0, \overline{\mu}_1, \ldots, \overline{\mu}_{N-1}\}$ is determined by the approximate cost-to-go functions $\tilde{J}_1, \tilde{J}_2, \ldots, \tilde{J}_N$ (given either by their functional form or by an algorithm to calculate their values at states of interest). Note that if the problem is stationary, and the functions $\tilde{J}_{k+1}$ are all equal, as they would normally be in an infinite horizon context, the policy $\overline{\pi}$ is stationary.

There are several alternative approaches for selecting or calculating the functions $\tilde{J}_{k+1}$. We distinguish two broad categories:

(1) *Explicit cost-to-go approximation*. Here $\tilde{J}_{k+1}$ is computed *off-line* in one of a number of ways. Some important examples are as follows:
   (a) By solving (optimally) a related simpler problem, obtained for example by state aggregation or by some other type of problem simplification, such as some form of enforced decomposition. The functions $\tilde{J}_{k+1}$ are derived from the optimal cost-to-go functions of the simpler problem. We will not discuss this approach further in this paper, and we refer to the author's textbook [13] for more details.
   (b) By introducing a parametric approximation architecture, such as a neural network or a weighted sum of basis functions or features. The idea here is to approximate the optimal cost-to-go $J_{k+1}(x)$ with a function of a given parametric form $\tilde{J}_{k+1}(x) = \hat{J}_{k+1}(x, r_{k+1})$, where r_{k+1} is a parameter vector. This vector is tuned by some form of *ad hoc*/heuristic method (as for example in computer chess) or some systematic method (for example, of the type provided by the neuro-dynamic programming and reinforcement learning methodologies, such as temporal difference and Q-learning methods; see the monographs by Bertsekas and Tsitsiklis [8], and Sutton and Barto [SuB98], and the recent edited volume by Barto et al. [2], which contain extensive bibliographies). There has also been considerable recent related work based on linear programming, actor-critic, policy gradient and other methods (for a sampling of recent work, see de Farias and Van Roy [18,19], Konda [30], Konda and Tsitsiklis [29], Marbach and Tsitsiklis [36], and Rantzer [41], which contain many other references). Again, this approach will not be discussed further in this paper.

(2) *Implicit cost-to-go approximation*. Here the values of $\tilde{J}_{k+1}$ at the states $f_k(x_k, u_k, w_k)$ are computed *on-line* as needed, via some computation of future costs, starting from these states (optimal or suboptimal/heuristic, with or without a rolling