since for each time t and each state $\mathbf{x} \in S_t$, we have to search U_t for the control that maximizes the right-hand side of (D.2).

The usual difficulty with dynamic programming in practice is that the state space S_t can become quite large, making the recursion above computationally complex. For example, in a RM problem with n inventory classes, each with capacities in the range $0, 1, \ldots, C$, the size of the state space is C^n. For even moderate values of C and n, this becomes prohibitively large. This "curse of dimensionality" is the main drawback to dynamic programming. However, for problems with a moderate state space, dynamic programming provides a general procedure for computing and analyzing optimal decisions.

Systems with Observable Disturbances

We next consider a variation of this traditional dynamic programming formulation that helps simplify many RM models. Specifically, consider a case in which we can base our control action $\mathbf{u}$ on perfect knowledge of the disturbance $\mathbf{w}(t)$. In other words, we allow the control to be a function of both the state $\mathbf{x}$ and the disturbance $\mathbf{w}(t)$, so that $\mathbf{u} = \mathbf{u}(\mathbf{x}, \mathbf{w}(t))$. The idea here is that in such systems, we can observe the disturbance before making our control decision and therefore base our decision on the realized value of $\mathbf{w}(t)$.

In this case, the basic dynamic programming recursion becomes

$$V_t(\mathbf{x}) = \max_{\{\mathbf{u}(\mathbf{x},\mathbf{w}(t)) \in U_t(\mathbf{x})\}} E\left[g_t(\mathbf{x}, \mathbf{u}(\mathbf{x}, \mathbf{w}(t)), \mathbf{w}(t)) + V_{t+1}(\mathbf{f}_t(\mathbf{x}, \mathbf{u}(\mathbf{x}, \mathbf{w}(t)), \mathbf{w}(t)))\right],$$

where $\mathbf{u}(\mathbf{x}, \mathbf{w}(t))$ emphasizes that we can select a different control $\mathbf{u}$ for each value of $\mathbf{w}(t)$. However, since we can choose a control based on knowing $\mathbf{w}(t)$, the above recursion can be rewritten as

$$V_t(\mathbf{x}) = E\left[\max_{\{\mathbf{u} \in U_t(\mathbf{x})\}} \{g_t(\mathbf{x}, \mathbf{u}, \mathbf{w}(t)) + V_{t+1}(\mathbf{f}_t(\mathbf{x}, \mathbf{u}, \mathbf{w}(t)))\}\right]. \qquad \text{(D.3)}$$

The recursion (D.3) can in fact be represented in the traditional form by expanding the state space. First, reindex the disturbances so that we have a new sequence of disturbance terms

$$\tilde{\mathbf{w}}(t) = \mathbf{w}(t+1), \quad t = 1, \ldots, T-1.$$

Consider adding the new system-state variable $\mathbf{y}(t)$, which along with $\mathbf{x}(t)$ is updated by the system equations

$$\begin{aligned} \mathbf{y}(t+1) &= \tilde{\mathbf{w}}(t) \\ \mathbf{x}(t+1) &= \mathbf{f}_t(\mathbf{x}(t), \mathbf{u}(t), \mathbf{y}(t)), \end{aligned}$$

where $\mathbf{f}_t$ is the same function as in (D.3). The initial state is $(\mathbf{x}(1), \mathbf{y}(1)) = (\mathbf{x}, \mathbf{w}(1))$ and the traditional dynamic programming recursion is

$$V_t(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{u} \in U_t(\mathbf{x})} E\left[g_t(\mathbf{x}, \mathbf{u}, \mathbf{y}) + V_{t+1}(\mathbf{f}_t(\mathbf{x}, \mathbf{u}, \mathbf{y}), \tilde{\mathbf{w}}(t))\right],$$

for all $\mathbf{x} \in S_t$ and all $\mathbf{y} \in W_t$. To see this can be converted to the same form as (D.3), define

$$G_t(\mathbf{x}) = E[V_t(\mathbf{x}, \tilde{\mathbf{w}}(t))],$$