DYNAMIC PROGRAMMING

P. R. Kumar
Department of ECE, University of Illinois, USA

Keywords: dynamic programming, running cost, one-step cost, terminal time, terminal cost, invariant embedding, principle of optimality, cost-to-go function, value function, Hamilton-Jacobi-Bellman partial differential equation, Markov decision processes, Markov policy, stationary policy, total cost, discounted cost, long-term average cost, long-run average cost, ergodic cost, average cost, positive dynamic programming, negative dynamic programming, value iteration algorithm, policy iteration algorithm, Poisson equation, differential cost function.

Contents

1. An Example to Illustrate the Dynamic Programming Method
2. Finite Horizon Discrete Time Deterministic Systems
2.1 Extensions
3. Finite Horizon Continuous Time Deterministic Systems
4. Time Varying Systems
5. Finite Horizon Discrete Time Stochastic Systems
6. Infinite Horizon Cost Functions
7. The Total Cost over an Infinite Horizon
8. The Discounted Cost Problem
9. The Average Cost Problem
10. Continuous Time Stochastic Systems
Glossary
Bibliography
Biographical Sketch

UNESCO – EOLSS SAMPLE CHAPTERS

Summary

Dynamic programming is a methodology for determining an optimal policy and the optimal cost for a multistage system with additive costs. It can be used in a deterministic or a stochastic environment, for a discrete time or a continuous time system, and over a finite time horizon or an infinite time horizon.

1. An Example to Illustrate the Dynamic Programming Method

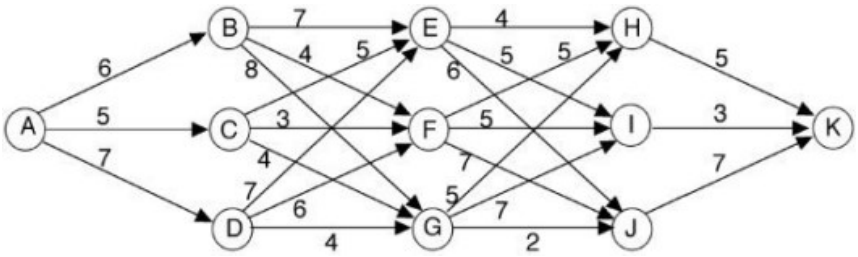

Figure 1: Example to introduce the dynamic programming method.

To understand the essence of dynamic programming, consider the example shown in Figure 1. There are 11 cities, labeled *A*, *B*, ..., *K*, and a network of one-way roads connecting them.

The length of each road connecting two cities is shown alongside it. It is desired to find the shortest path from city *A* to city *K*.

The dynamic programming method solves this problem as follows. Consider cities *H*, *I*, and *J*. From each of them the shortest path to city *K* is shown in Figure 2. This is obvious since from each of them there is only one path to city *K*. The shortest distances from the cities *H*, *I*, and *J* to *K* are respectively, 5, 3, and 7, and these are shown alongside the cities in Figure 2.

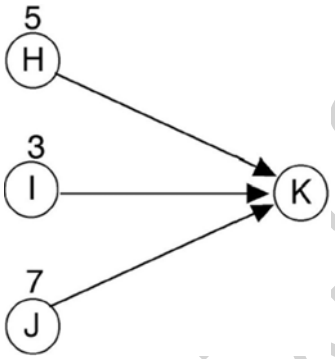

Figure 2: The shortest paths to K from H, I, and J.

Now turn to cities *E*, *F*, and *G*. Consider city *E* first. There are 3 roads, one going "high" to *H*, one going "medium" to *I*, and the other going "low" to *J*. If one takes the high road to *H*, then the distance traveled to *H* is 4, from which point the remaining shortest distance to *K* (already computed in Figure 2) is 5, giving a total distance to *K* of 9; see Figure 3. If however one takes the "medium" road to *I* from *E*, then the distance traveled to *I* is 5, from which the shortest distance to *K* (from Figure 2) is 3, giving a total distance to *K* of 8. Finally, if one takes the low road from *E* to *J*, then the distance to *J* is 6, from which the remaining shortest distance to *K* is 7, yielding a total distance of 13. Thus the shortest path from *E* to *K* is via *I*, and this shortest distance is 8.

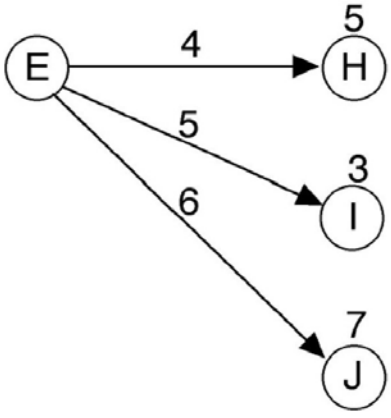

Figure 3: Determining the shortest path from E to K.

UNESCO – EOLSS SAMPLE CHAPTERS

In computing the shortest distance from *E* to *K*, we have thus used the following equation:

$$\text{Shortest distance from E to K} = \text{Min}\left\{\begin{array}{ccc} \text{Distance from E to H} & \text{Distance from E to I} & \text{Distance from E to J} \\ + & + & + \\ \text{Shortest distance from H to K} & \text{Shortest distance from I to K} & \text{Shortest distance from J to K} \end{array}\right\}. \quad (1)$$

In a similar way, we can calculate the shortest paths from *F* and *G* to *K*. We can determine that the shortest path from *F* to *K* is via *I* and the shortest distance is 8. Also, the shortest path from *G* to *K* is via *J*, and the shortest distance is 9. These results are shown in Figure 4, where the shortest distance to *K* is shown alongside each city, and the shortest paths are indicated.

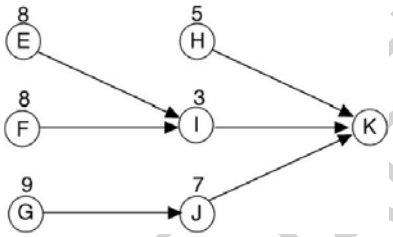

Figure 4: The shortest paths and distances from E, F, and G.

Now we turn to cities *B*, *C*, and *D*. Repeating the procedure, we find the shortest paths and distances from *B*, *C*, and *D* to *K* as shown in Figure 5.

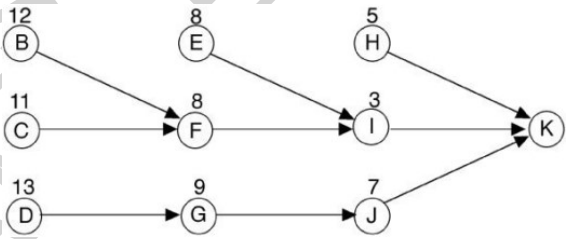

Figure 5: The shortest paths and distances from B, C, and D.

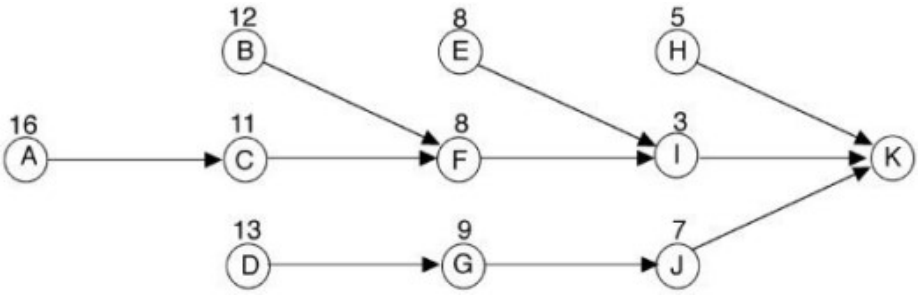

Figure 6: Shortest path and distance from A.

Finally, repeating the procedure from node *A*, we determine that the shortest path from *A* to *K* is via *C*, *F*, and *I*, and the shortest distance is 16, as shown in Figure 6.

There are several observations that we can make about the method we have used:

(i) The cost function of interest was stage-wise additive. That is, the total distance is the sum of distances of each link traversed.

(ii) In calculating the shortest paths from *A* to *K*, we have actually calculated the shortest path from *every* node *A*, *B*, *C*, ..., *J* to *K*, as shown in Figure 6. Technically, we have "embedded" our original problem of finding the shortest path from *A* to *K* into the family of problems of determining the shortest path from every node to *K*. Because of this property, dynamic programming is sometimes called "the method of invariant embedding".

(iii) Because we have calculated the shortest path from every node to *K*, dynamic programming can be computationally very intensive.

(iv) We solved the problem *backwards* by starting at the *end* from nodes *G*, *H*, and *I*. Thus dynamic programming calculates optimal actions backwards in time.

(v) In solving the problem, we have used the property that the shortest distance from any node to *K* is the minimum, over the next node that can be reached, of the distance to the next node plus the shortest distance from that next node to *K*; see Eq. (1). We see that if the shortest path from *A* to *K* is $A \to C \to F \to I \to K$, then the shortest path from *F* to *K* is $F \to I \to K$. More generally, we have the property that tail segments of optimal paths are themselves optimal. This is an example of what is called *The Principle of Optimality*.

(vi) The recursion in Eq. (1) can be written more generally as:

$$\begin{array}{c}\text{Optimal cost}\\ \text{from x}\end{array} = \min_{\text{Immediate action u}} \left\{ \begin{array}{c}\text{Immediate cost}\\ \text{of taking action}\\ \text{u in state x}\end{array} + \begin{array}{c}\text{Optimal cost}\\ \text{from the state}\\ \text{reached by taking}\\ \text{action u in state x}\end{array} \right\}. \quad (2)$$

This is called the *dynamic programming equation.*

(vii) Dynamic programming calculates the optimal decision from each node. That is, it calculates the optimal action as a function of what the current state is. For example, the optimal action in state *G* is to go low. Hence dynamic programming yields the optimal policy mapping states to actions. It specifies the final result as an optimal *state feedback policy* rather than as an optimal open loop policy.

(viii) From (vii) it follows that dynamic programming can be applied for determining actions when the "state" of a system is known or can be observed.

Armed with these observations, we can address more general situations.

2. Finite Horizon Discrete Time Deterministic Systems

Consider the discrete time controlled dynamic system:

$$x(t+1) = f(x(t), u(t)) \quad (3)$$

UNESCO – EOLSS SAMPLE CHAPTERS

where

$$u(t) \in U. \tag{4}$$

The set U is called the *constraint set*. Consider the cost function

$$\sum_{t=0}^{T} c(x(t), u(t)). \tag{5}$$

Starting from a given initial state $x(0)$, it is desired to determine a control policy which minimizes the total cost Eq. (5). The function $c(x, u)$ is called the *running cost* or *one-step cost function*. The ending time T is called the *terminal time*.

Let us define

$V(x, t)$ = optimal *remaining* cost when the system is in state x at time t.

This function is called the *optimal cost-to-go* function or the *value function*.

At time $T + 1$ there is no remaining cost. Hence,

$$V(x, T+1) \equiv 0. \tag{6}$$

Proceeding backwards from time T, we have the recursion:

$$V(x,t) = \text{Min}_{u \in U} \{c(x,u) + V(f(x,u), t+1)\}, \tag{7}$$

which is a generalization of Eq. (2). (If the "Min" in Eq. (7) is not attained one uses an "Inf," in which case an optimal policy does not exist, but there exists one which is arbitrarily close to optimal.)

It can be shown by a contradiction argument that $V(x, t)$ is indeed the optimal remaining "cost-to-go" from state x and time t. That is, the validity of the Principle of Optimality can be established.

Let $u(x, t)$ attain the "Min" in Eq. (7). It can be shown that $u(x, t)$ is an optimal policy. It is expressed as a state feedback control policy. The optimal cost starting from the initial condition x at time 0 is $V(x, 0)$.

UNESCO – EOLSS SAMPLE CHAPTERS

-
-
-

TO ACCESS ALL THE **18 PAGES** OF THIS CHAPTER,
Click here

Bibliography

Bellman R. (1957). *Dynamic Programming*, 342 pp. Princeton, NJ: Princeton University Press. [A good expose of dynamic programming.]

Bertsekas D.P. (1987). *Dynamic Programming*: *Deterministic and Stochastic Models*, 376 pp. Englewood Cliffs, NJ: Prentice-Hall. [A comprehensive account of dynamic programming in discrete-time.]

Bertsekas D.P. and Shreve S.E. (1978). *Stochastic Optimal Control*: *The Discrete Time Case*, 323 pp. New York, NY: Academic Press. [A good reference for handling technical issues related to measurability of functions.]

Blackwell D. (1962). Discrete Dynamic Programming. *Ann. of Math. Statist.*, **33**, 719-726. [A good original reference for results on the long-term average cost problem.]

Blackwell D. (1965). Discounted Dynamic Programming. *Ann. of Math. Statist.*, **36**, 226-335. [An original reference.]

Blackwell D. (1965). Positive Dynamic Programming. *Proceedings of the* 5*th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 415-418. [An original reference.]

Fleming W.H. and Rishel R.W. (1975). *Deterministic and Stochastic Optimal Control*, 222 pp. Berlin: Springer-Verlag. [A good reference for continuous time stochastic control.]

Kumar P.R. and Varaiya P.P. (1986). *Stochastic Systems*: *Estimation, Identification and Adaptive Control*, 358 pp. Englewood Cliffs, NJ: Prentice-Hall, 1986. [Contains a concise treatment of dynamic programming, including the case of partial observations.]

Strauch R. (1966). Negative Dynamic Programming," *Ann. Math. Statist.*, **37**, 871-890. [An original reference.]

Biographical Sketch

P. R. Kumar is the Franklin W. Woeltge Professor of Electrical and Computer Engineering, and a Research Professor in the Coordinated Science Laboratory, at the University of Illinois, Urbana-Champaign. He was the recipient of the Donald P. Eckman Award of the American Automatic Control Council. He has presented plenary lectures at the SIAM Annual Meeting and the SIAM Control Conference in 2001, the IEEE Conference on Decision and Control in San Antonio, Texas, 1993, the SIAM Conference on Optimization in Chicago, 1992, the SIAM Annual Meeting at San Diego, 1994, Brazilian Automatic Control Congress, and the Third Annual Semiconductor Manufacturing, Control and Optimization Workshop. He is a co-author with Pravin Varaiya of the book, *Stochastic Systems: Estimation, Identification and Adaptive Control* He serves on the editorial boards of Communications in Information and Systems, Journal of Discrete Event Dynamic Systems; Mathematics of Control Signals and Systems; Mathematical Problems in Engineering: Problems, Theories and Applications; and in the past has served as Associate Editor at Large for IEEE Transactions on Automatic Control; Associate Editor of SIAM Journal on Control and Optimization; Systems and Control Letters; Journal of Adaptive Control and Signal Processing; and the IEEE Transactions on Automatic Control. He is a Fellow of IEEE. Professor Kumar's current research interests are in wireless networks, distributed real-time systems, wafer fabrication plants, and machine learning.

UNESCO – EOLSS SAMPLE CHAPTERS