学术论文

动态规划的正向递推方法

张 钊[1,2]，裴燕玲[2]，张仁宝[1]
（1. 天津大学电气与自动化学院，天津 300072；2. 山东省建筑设计研究院，济南 250001）

[摘要] 动态规划求最优解是一个反向递推的求解过程，以实例为依据，用正向递推的方法求解动态规划的最优值，并推出动态规划的基本方程和密尔顿－雅可比方程，是对动态规划求最优解方法的探讨。利用动态规划的正向递推方法，在应用中可以大大减少计算量，扩大了它的应用范围。

[关键词] 动态规划；多级决策；泛函；最优值
[中图分类号] O221.3 [文献标识码] A [文章编号] 1009－1742（2005）02－0062－04

1 引言

20 世纪 50 年代初，美国数学家贝尔曼提出了最优性原理和动态规划方法，它和前苏联数学家庞里亚特金同期提出的最大值原理共同构成现代控制理论的重要的定理[1]，引起了控制理论的重大变革。从此、控制理论由频域法转为时域法；由经典控制理论时代进入现代控制理论时代。

动态规划现今在许多技术领域，特别是在管理科学、运筹学、经济学等方面得到了广泛的应用，它是处理控制变量限制在一定闭集内的最优控制问题的有效的数学方法，它把复杂的控制问题变为多级决策过程的递推函数关系[2,3]，其基础和核心是最优原理，动态规划的求解过程是一个反向递推的求解过程，很多文献中有详尽的求解过程[4,5]，近几年，人们从各方面对动态规划的求解过程进行了探讨，取得了一定的成果[6~8]，由于求解过程是反向递推的，从终端开始，所以在多级决策过程中，无论实际解决的问题是多少，都要求解全部过程，这使动态规划的计算量大大增加，限制了它的应用，笔者用正向递推方法求解动态规划的最优值，并推导出哈密尔顿－雅可比方程，是对动态规划求最优解方法的探讨。

2 多级决策的正向求解

多级决策是指把一个过程按时间或空间顺序分成若干级，然后对每一级做出决策，以使整个过程取得最优结果。动态规划是解决多级决策过程优化问题的强有力的工具，传统动态规划的求解方法是从终端开始，用反向递推的办法。下面以行车问题为例，说明动态规划的正向求解过程。

设汽车从 A 城出发到 B 城，途中需经过三条河流，它们各有两座桥 P，Q 可供选择通过，如图 1，各段间的行车时间（h）见图中标注，现在确定一条最优路线，使从 A 城到 B 城的行车时间最短。

动态规划正向求解的原则——所选择的最优路线必须保证其前部子路线是最优的。在图 1 中，如果 $AQ_1P_2Q_3B$ 是最优路线，那么从起始点到这条路线上任一中间点的子路线必定也是最优的，否则 $AQ_1P_2Q_3B$ 就不是最优路线了。

根据这一原则求解最优路线，从起点开始，按时间最短为目标，逐步向后，求出起点至各中间站的最优值，直至求出至终点的最优值。

第一段，起点 A 的后一站是 P_1，Q_1，无论汽

[收稿日期] 2004－04－15；修回日期 2004－05－19
[作者简介] 张 钊（1965－），男，山东济南市人，天津大学自动化学院博士研究生，山东省建筑设计研究院高级工程师

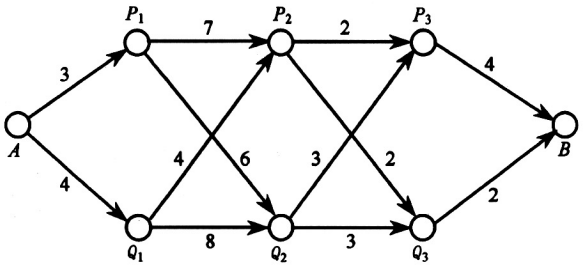

图 1　段间行车时间

Fig.1　Running time between segments

车到终点 B 的路径如何，总要经过 P_1 或 Q_1，到 P_1 和 Q_1 都只有一条路线，所以到 P_1 的最优路线是 AP_1，历时 3 h；到 Q_1 的最优路线是 AQ_1，历时 4 h。

第二段，P_1，Q_1 的后一站是 P_2，Q_2，到 P_2 有两条路线，从 P_1 到 P_2——AP_1P_2，历时 10 h；从 Q_1 到 P_2——AQ_1P_2，历时 8 h，其最短历时 8 h。到 Q_2 也有两条路线，从 P_1 到 Q_2——AP_1Q_2，历时 9 h；从 Q_1 到 Q_2——AQ_1Q_2，历时 12 h，其最短历时 9 h。

第三段，P_2，Q_2 的后一站是 P_3，Q_3，到 P_3 有两条路线，从 P_2 到 P_3，要使历时最短，从 A 到 P_2 一定走路线 AQ_1P_2，即 $AQ_1P_2P_3$，历时 10 h；从 Q_2 到 P_3，要使历时最短，从 A 到 Q_2 一定走路线 $AP1Q_2$，即 $AP_1Q_2P_3$，历时 12 h，其最短历时 10 h。到 Q_3 也有两条路线，从 P_2 到 Q_3，要使历时最短，从 A 到 P_2 一定走路线 AQ_1P_2，即 $AQ_1P_2Q_3$，历时 10 h；从 Q_2 到 Q_3，要使历时最短，从 A 到 Q_2 一定走路线 AP_1Q_2，即 $AP_1Q_2Q_3$，历时 12 h，其最短历时 10 h。

第四段，P_3，Q_3 的后一站是终点站 B，到终点站 B 有两条路线，从 P_3 到 B，要使历时最短，从 A 到 P_3 一定走路线 $AQ_1P_2P_3$，即 $AQ_1P_2P_3B$，历时 14 h；从 Q_3 到 B，要使历时最短，从 A 到 Q_3 一定走路线 $AQ_1P_2Q_3$，即 $AQ_1P_2Q_3B$，历时 12 h，其最短历时 12 h。

所以最优路线是 $AQ_1P_2Q_3B$，最优值是 12 h。图 2 为正向递推求解的图示过程。

动态规划传统的反向递推求解原则：所选择的最优路线必须保证其后部子路线是最优的。在图 2 中，如果 $AQ_1P_2Q_3B$ 是最优路线，那么这条路线上从任一中间点到终点的子路线必定也是最优的，否则 $AQ_1P_2Q_3B$ 就不是最优路线了。求解过程是从终点开始，按时间最短为目标，逐步向前，求出各中间点到终点的最优值，直至求出起始点至终点的最优值。详见文献[4, 5]，图 3 是传统反向递推求解的图示过程，从图 3 中可以看出：传统反向递推求解与正向递推求解其结果完全一致。

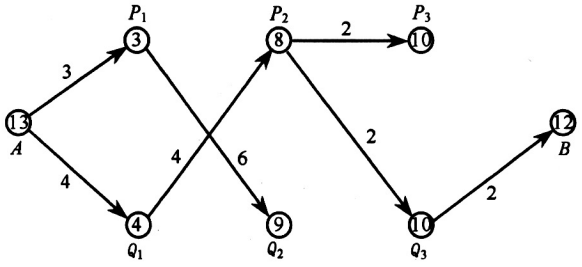

图 2　正向递推求解过程

Fig.2　Recursion solution process in positive direction

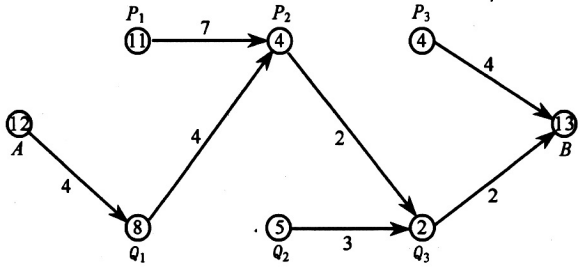

图 3　反向递推求解过程

Fig.3　Recursion solution process in reverse direction

3　离散系统动态规划的正向递推方法

设离散系统（见图 4）的状态方程为

$$x_{k+1} = f[x_k, u_k] \quad k = 0, 1, \cdots, N-1$$

式中 $x_{k+1} = x(k+1)$ 为 n 维状态矢量在 $(k+1)T$ 时刻的值，$u_k = u(k)$ 为 r 维控制矢量在 kT 时刻的值，f 为 n 维矢量函数。

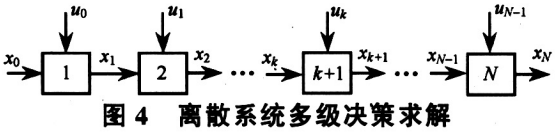

图 4　离散系统多级决策求解

Fig.4　Multistage decision solution to discrete systems

状态初值　　$x(0) = x_0$，

控制约束　　$g[x_k, u_k] \leqslant 0$，

性能泛函　　$J = \Phi[x_N] + \sum_{k=0}^{N-1} L[x_k, u_k]$，

式中 $\Phi[x_N]$ 是对终端状态 $x_N = x(N)$ 的要求。

寻求一个最优控制序列 $\{u_k^*\}$ $(k = 0, 1, \cdots, N-1)$，使上述性能泛函取极值。

根据多级决策的正向求解的方法，N 段最优决策过程中，前 k 段决策一定是最优决策。

$$J_k^*(x_k) = \min\{J_{k-1}^*[x_{k-1}] + L[x_{k-1}, u_{k-1}]\}$$

$$x_k = f[x_{k-1}, u_{k-1}],$$

即 $J_0^*(x_0) = \Phi[x_N]$,

$$J_1^*(x_1) = \min\{J_0^*[x_0] + L[x_0, u_0]\}$$

$$x_1 = f[x_0, u_0],$$

$$\vdots$$

$$J_N^*(x_N) = \min\{J_{N-1}^*[x_{N-1}] + L[x_{N-1}, u_{N-1}]\}$$

$$x_k = f[x_{k-1}, u_{k-1}]。$$

对于一个多段决策的最优求解过程，其初始值和终端状态都是给定的，利用正向求解，很容易求出前 k 段的最优值。

4 连续系统动态规划的正向递推方法

设连续系统（见图5）的状态方程为

$$\dot{x} = f[x, u, t],$$

状态初值 $x(t_0) = x_0$,

终端约束 $N[x(t_f), t_f] = 0$,

性能泛函 $J[x,t] = \Phi[x(t_f)] + \int_0^{t_f} L[x,u,t]dt$,

寻求一个最优控制 $u^*(t)$，使上述性能泛函取极值。

根据多级决策正向求解的方法，如果 $x^*(t)$ 是以 $x(t_0)$ 为初始状态，以 $x(t_f)$ 为终端状态的最优轨线，那么以 $x(t')$ 为终端状态的前半段一定是最优轨线。

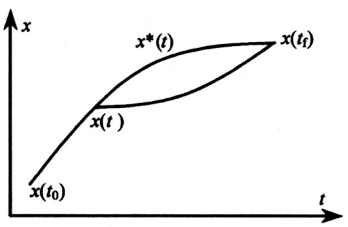

图5 连续系统多级决策求解

Fig.5 Multistage decision solution to continuous systems

若取 $t_f = t, t' = t_f - \Delta t$ 则最优性能泛函

$$J^*[x(t),t] = \min\left\{\Phi[x(t)] + \int_{t_0}^{t} L[x,u,t]dt\right\} =$$

$$\min\left\{\Phi[x(t)] + \int_{t_0}^{t-\Delta t} L[x,u,t]dt + \int_{t-\Delta t}^{t} L[x,u,t]dt\right\},$$

$$J^*[x(t-\Delta t),t-\Delta t] = \min\left\{\Phi[x(t)] + \int_{t_0}^{t-\Delta t} L[x,u,t]dt\right\}。$$

当 Δt 很小时

$$\int_{t-\Delta t}^{t} L[x,u,t]dt \approx L[x, u, t]\Delta t,$$

$$x(t-\Delta t) = x - \Delta t\,dx/dt + \cdots =$$

$$x - \Delta x + \cdots \approx x - f[x, u, t]\Delta t,$$

$$J^*[x(t-\Delta t), t-\Delta t] \approx J^*[x-\Delta x, t-\Delta t] = J^*[x(t), t] - \left[\frac{\partial J^*(x,t)}{-\partial x}\right]^{\mathrm{T}}\Delta x - \frac{\partial J^*(x,t)}{-\partial t}\Delta t。$$

所以

$$J^*[x(t), t] = \min\{J^*[x(t-\Delta t), t-\Delta t] + L[x, u, t]\Delta t\} = \min\left\{J^*[x(t),t] - \left[\frac{\partial J^*(x,t)}{-\partial x}\right]^{\mathrm{T}}\Delta x - \frac{\partial J^*(x,t)}{\partial x}\Delta t + L[x,u,t]\Delta t\right\} =$$

$$J^*[x(t), t] - \frac{\partial J^*(x,t)}{-\partial t}\Delta t + \min\left\{-\left[\frac{\partial J^*(x,t)}{-\partial x}\right]^{\mathrm{T}} f[x,u,t]\Delta t + L[x,u,t]\Delta t\right\},$$

整理得

$$-\frac{\partial J^*(x,t)}{\partial t} = \min\left\{L[x,u,t] + \left[\frac{\partial J^*(x,t)}{-\partial x}\right]^{\mathrm{T}} f[x,u,t]\right\},$$

这就是连续系统动态规划基本方程，即贝尔曼方程。

令哈密尔顿函数

$$H[x, u, \boldsymbol{\lambda}, t] = L[x, u, t] + \left[\frac{\partial J^*(x,t)}{\partial x}\right]^{\mathrm{T}} f[x,u,t] = L[x, u, t] + \boldsymbol{\lambda}^{\mathrm{T}} f[x, u, t] \quad \boldsymbol{\lambda} = \frac{\partial J^*(x,t)}{\partial x},$$

得密尔顿－雅可比方程

$$-\frac{\partial J^*(x,t)}{\partial t} = H[x, u, \boldsymbol{\lambda}, t]$$

5　结论

笔者以实例说明为依据，用正向递推的方法求解动态规划的最优值，并推出动态规划的基本方程和密尔顿－雅可比方程，是对动态规划求最优解方法的探讨。利用动态规划的正向递推方法，在应用中不仅可以大大减少计算量，而且使一些很难利用反向递推求解的问题得到解决，扩大了动态规划的应用范围。在实际技术领域中，尤其对于离散的控制系统，如何充分的利用动态规划的正向递推方法，简化最优值的求解过程，解决传统动态规划方法难以解决的问题，还有待于进一步探讨。

参考文献

[1] Acquot R G. Modern Digital Control System [M]. Marcel: Dekker Inc, 1981

[2] Burghes D, Graham A. Introduction to Control, Including Optimal Control [M]. Ellis: Horword Limited, 1980

[3] Csaki F. Modern Control Theories Nonlinear Optimal and Adaptive System [M]. Akademial: Kiado, 1975

[4] 刘　豹. 现代控制理论 [M]. 北京：机械工业出版社，1996

[5] 胡寿松. 自动控制原理 [M]. 北京：国防工业出版社，2000

[6] Sadr J, Malhame R P. Decomposition/aggregation-based dynamic programming optimization of partially homogeneous unreliable transfer lines [J]. IEEE Transactions on Automatic Control, 2004, 49 (1): 68～81

[7] Kossmann D, Stocker K. Iterative dynamic programming: a new class of query optimization algorithms [J]. ACM Transactions on Database Systems, 2000, 25 (1): 43～82

[8] Liu Y A, Stoller S D. Dynamic programming via static incrementalization [J]. Higher-Order and Symbolic Computation, 2003, 16 (1, 2): 37～62

The Forward Recurrent Method for Dynamic Programming

Zhang Zhao[1,2], Pei Yanling[2], Zhang Renbao[1]

(1. *School of Electrical and Automation Engineering, Tianjin University, Tianjin* 300072, *China*;
2. *Shandong Provincial Architectural Design & Research Institute, Jinan* 250001, *China*)

[**Abstract**] Backward recurrent method is usually adopted in seeking optimal solution by dynamic programming. A forward recurrent method to find optimal solution by dynamic programming is presented on the basis of an instance. The fundamental equation of dynamic programming and Millton-Jacobi's equation are also derived. It's an exploratory study on optimal solution of dynamic programming. An amount of work is reduced in calculating by using forward recurrent method, while applied range of the method is expanded.

[**Key words**] dynamic programming; multi-level decision; functional equation; optimal solution