强化学习与最优控制（样章）[1]

德梅萃 · P. 博赛卡斯（Dimitri P. Bertsekas）著
李宇超（Yuchao Li）译

[1]清华大学出版社，责任编辑：王一玲

第一章　精确动态规划

在本章中，我们介绍精确动态规划（简称 DP）的一些背景知识，同时着眼于本教材的主题，即一些次优的求解方法。这些方法通常被冠以几个不同的、但本质上等价的名称：强化学习（*reinforcement learning*）、近似动态规划（*approximate dynamic programming*）和神经元动态规划（*neuro-dynamic programming*）。在本教材中，作为这些方法的统称，我们主要使用其中最通俗的名字：强化学习（简称 RL）。

我们首先介绍包含有限多的连续决策阶段的问题，即所谓有限阶段问题。因为只涉及有限多的决策阶段，该类问题在理解与分析中相对容易。对于更加复杂的无穷阶段问题的讲解则会在第 4 章到第 6 章给出。此外，我们分开介绍确定性问题和随机问题（分别对应 1.1 节和 1.2 节）。这样安排是由于相较随机问题，确定性问题更加简单，因而更适于作为学习最优控制理论的切入点。此外与随机问题所相比，确定性问题还具有一些特有的利于求解的特点。这些特点有助于我们通过更加多样的方式来求解。例如，当用于解决确定性问题时，基于仿真的方法可以得到极大的简化，而且也更容易理解。

最后，我们在 1.3 节提供多种动态规划建模实例，用以阐明 1.1 节和 1.2 节中的一些概念。熟悉动态规划理论的读者可选择在粗读 1.3 节后就跳过本章的其他部分，直接开始第 2 章的学习。我们关于近似动态规划方法的讲解也将在第 2 章展开。

1.1　确定性动态规划

所有动态规划问题都会涉及一个离散时间的动态系统。在控制的影响下，该系统会相应地生成一个由状态构成的序列。在有限阶段问题中，系统会经历 N 个时刻（也称为阶段）的演化。在 k 时刻，系统的状态和所受到的控制分别记作 x_k 和 u_k。在确定性系统中，x_{k+1} 不是随机生成的，即它由 x_k 和 u_k 完全决定。

1.1.1　确定性问题

一个确定性动态规划问题涉及如下离散时间的动态系统

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

其中，

k 是时刻；

x_k 是系统状态，属于某种空间的一个元素；

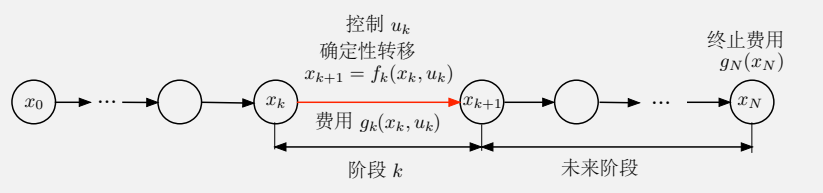

图 1.1.1: 一个确定性的 N 阶段的最优控制问题。从 x_k 出发，在控制信号 u_k 的作用下，下一时刻的状态依据以下系统方程

$$x_{k+1} = f_k(x_k, u_k)$$

完全确定，并产生一个取值为 $g_k(x_k, u_k)$ 的阶段费用。

u_k 是控制或决策变量，它属于某个给定的集合 $U_k(x_k)$，该集合由 x_k 决定，并可随时刻 k 而变化；

f_k 是关于 (x_k, u_k) 的函数，用于描述系统状态从 k 时刻到 $k+1$ 时刻的演化机制；

N 是时域或决策可作用于系统的时刻的总数。

我们称所有可能的 x_k 组成的集合为 k 时刻的状态空间（*state space*）。该集合可以是任意类型的集合，并且可以随着时刻 k 而发生改变；状态空间所囊括的情况的一般性正是动态规划方法的极大的优点之一（*this generality is one of the great strengths of the DP methodology*）。类似地，由所有可能的 u_k 构成的集合被称为控制空间（*control space*）。同理，U_k 可以是任意类型的集合，并且可能随着时刻 k 的不同而改变。

除上述概念外，动态规划问题还会涉及费用函数这一概念。我们记 k 时刻所产生的阶段费用为 $g_k(x_k, u_k)$，那么通过把阶段费用从当前时刻起累加起来就定义了当前状态的费用函数。规范地说，$g_k(x_k, u_k)$ 是一个关于 (x_k, u_k) 函数。该函数的输出为实数，且函数本身也可能随着时刻 k 而发生变化。对于一个给定的初始状态 x_0，控制序列 $\{u_0, \ldots, u_{N-1}\}$ 的总费用为

$$J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k), \tag{1.2}$$

其中 $g_N(x_N)$ 为产生于系统演化终点的终止费用。上述总费用从数学上看是定义良好的，因为控制序列 $\{u_0, \ldots, u_{N-1}\}$ 和初始状态 x_0 通过式(1.1)共同确定了状态序列 $\{x_1, \ldots, x_N\}$。我们希望在所有满足控制约束的决策序列 $\{u_0, \ldots, u_{N-1}\}$ 中最小化费用(1.2)，从而得到最优值[1]

$$J^*(x_0) = \min_{\substack{u_k \in U(x_k) \\ k=0,\ldots,N-1}} J(x_0; u_0, \ldots, u_{N-1})$$

定义的关于 x_0 的函数。图1.1.1说明了确定性问题的主要组成部分。

接下来，我们将介绍一些确定性问题的示例。

[1]在本书中我们全部使用 “min”（在应使用 “inf” 的位置）来表示在一个可行决策集上取得的最小值。即使当我们不确定上述最小值能否通过某一可行的决策取得时，我们仍会采用符号 “min”。

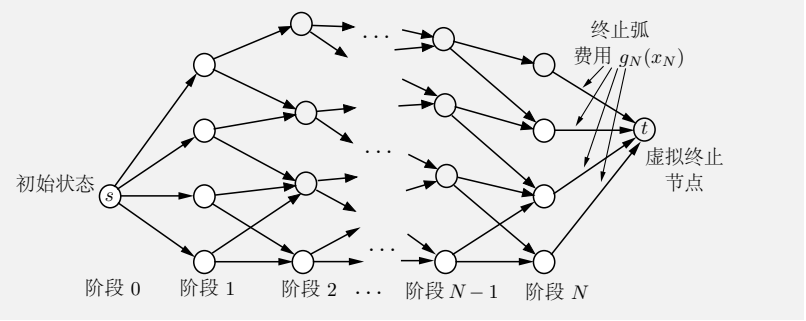

图 1.1.2: 一个确定性有限状态问题的转移图。图中的节点代表状态 x_k，弧则代表状态决策对 (x_k, u_k)。弧 (x_k, u_k) 以 x_k 和 $f_k(x_k, u_k)$ 作为起点和终点。我们把相应状态转移的费用 $g_k(x_k, u_k)$ 视作该弧的弧长。这个确定性有限状态的问题就等价于寻找从初始节点 s 到终止节点 t 的最短路径。

离散最优控制问题

许多情况下，状态与控制空间本身就是由有限多的离散元素构成的。在此类问题中，从任意 x_k 到可能的 x_{k+1} 的系统状态转移都可以很方便地用有向无环图来描述。图中的节点就代表了系统的状态，而弧则代表了状态动作对 (x_k, u_k)。以 x_k 为起点的每个弧对应于控制约束集 $U_k(x_k)$ 中的一个控制选项，并且其终点为下一时刻状态 $f_k(x_k, u_k)$，见图1.1.2。为了有效采用图来处理该问题的最终阶段，我们在图中添加一个虚拟终止节点 t。每个 N 阶段的状态 x_N 与终止节点 t 之间被一个费用为 $g_N(x_N)$ 的弧相连。

注意到每一个控制序列 $\{u_0, \ldots, u_{N-1}\}$ 都对应于以初始状态（阶段 0 的状态 s）为起点、而终止于 N 阶段的一个状态的一条路径。如果我们把一条弧对应的费用视作其长度，那么我们容易发现求解一个确定性的有限状态有限阶段问题就等价于寻找其相应转移图中以初始节点 s 为起点、以终止节点 t 为终点的最小长度（最短）的路径（*a deterministic finite-state finite horizon problem is equivalent to finding a minimum-length (or shortest) path from the initial node s of the graph to the terminal node t*）。此处路径的长度指的是构成路径的弧的长度之和。[2]

一般而言，组合优化问题可以用确定性、有限状态、有限阶段的最优控制问题来描述。我们用以下的调度问题来诠释这一观念。

例 1.1.1 (一个确定性调度问题)**.** 假设为了生产某一产品，在某一机器上需要执行四道工序。这四道工序以字母 A、B、C 和 D 来表示。我们假设，工序 B 需在工序 A 之后执行，工序 C 需在工序 D 之前执行。(因此，工序序列 CDAB 是允许的，而 CDBA 则不允许。) 从工序 m 到 n 的编排费用 C_{mn} 给定。此外，从工序 A 和 C 开始分别需要的启动费用是 S_A 和 S_C (见图*1.1.3*)。一个工序序列的费用即该序列对应的阶段费用之和；例如，工序序列 ACDB 的费用为

$$S_A + C_{AC} + C_{CD} + C_{DB}.$$

我们可以把该问题视作关于三个决策序列的问题，即安排执行前三道工序 (第四道工序由前三道确定)。因此，将已经安排的工序序列作为状态就是一个合理的选择，相应的，初

[2]事实上，(在包括有环的任何有向图中）求解最短路径问题都可以转化为有限状态的确定性最优控制问题，这部分内容我们将会在 1.3.1 节讲解。更多与本节内容关联的、关于最短路径求解方法的详细介绍，参见 [Ber17] 的 2.1 节，[Ber91]，[Ber98]。

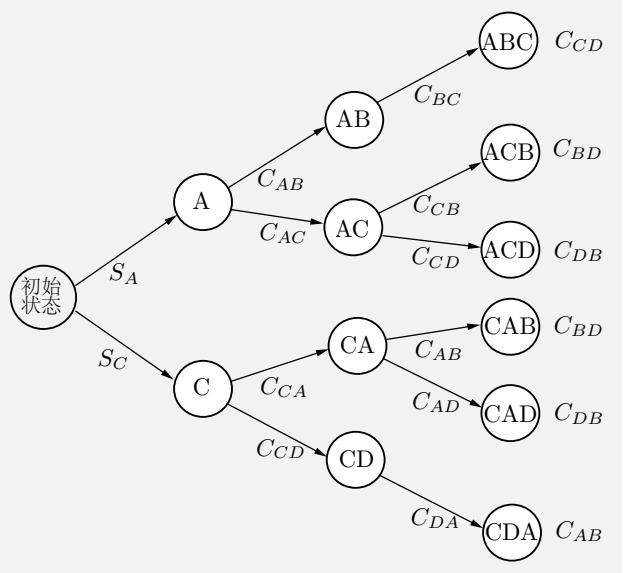

图 1.1.3: 例1.1.1所述的确定性调度问题的转移图。图中的每条弧都代表了把某个状态（弧的起点）导向另一状态（弧的终点）的一个决定。弧所对应的费用标注在其一旁。最终工序所对应的费用则作为终止费用标注在图中的终止节点的一旁。

始状态就是一个人为引入的状态，用来表示决策过程的开端。图*1.1.3*给出了该问题对应的状态及决策的所有可能的状态转移。此处的问题是确定性的，即在任意一个给定的状态下任选一个控制，系统下一时刻的状态都是唯一确定的。例如，在状态 AC 时决定安排工艺 D，那么下一时刻状态即为 ACD，且该决策花费 C_{CD} 的费用。因此，上述问题可以很方便地用图*1.1.3* 所示的转移图来说明。图中的每一条以初始状态为起点、以某一终止时刻的状态为终点的路径即为原问题的一个解。这一解的费用就是构成该路径的所有弧的费用与终止费用之和。该问题的最优解就对应于相应转移图中费用最小的上述路径。

连续空间最优控制问题

控制理论中的许多经典问题都涉及一个连续的状态空间。例如，状态空间可以是欧几里得空间，即由维度为某正整数 n 的向量构成的空间。这类问题中就包括了线性二次型问题（*linear-quadratic problems*）。线性二次型问题涉及线性的系统方程，二次型的费用函数，且不存在控制约束。下面我们就给出该类问题的一个示例。在我们所举的例子中，状态和控制都是一维的。但该类问题可以拓展到非常普遍的多维问题中（见 [Ber17]3.1 节）。

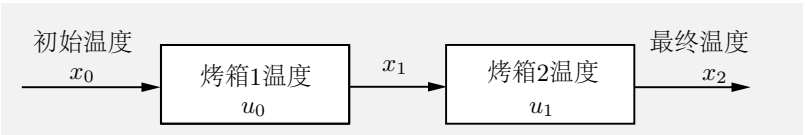

图 1.1.4: 当 $N = 2$ 时例1.1.2介绍的线性二次型问题。该材料温度根据系统方程 $x_{k+1} = (1 - a)x_k + au_k$ 演化。其中，a 为取值于开区间 $(0, 1)$ 的标量。

例 1.1.2 (一个线性二次型问题)**.** 某一材料需要依次通过 N 个烤箱（见图*1.1.4*）。记

x_0：该材料的初始温度，

x_k, $k = 1, \ldots, N$：该材料经过烤箱 k 之后的温度，

u_{k-1}, $k = 1, \ldots, N$：在烤箱 k 中作用于该材料的热能。在实际中，控制量 u_k 的值会受到某些约束，例如非负性。然而，为了便于理论分析，我们可以先考虑控制不受约束的情况，然后再检验所得的解是否满足相应问题中某些天然存在的约束。

我们假设系统方程如下

$$x_{k+1} = (1-a)x_k + au_k, \quad k = 0, 1, \ldots, N-1,$$

其中，a 是一个已知的位于开区间 $(0, 1)$ 的标量。我们的目标是使用相对较少的能量，使材料的最终温度 x_N 尽可能接近目标温度 T。我们用如下形式表示费用函数

$$r(x_N - T)^2 + \sum_{k=0}^{N-1} u_k^2,$$

其中 $r > 0$ 为一个给定标量，用于表示对上式的两部分费用，即所得最终温度与目标温度的误差以及所花费的总的加热能量，之间的权衡。

对于状态和控制不受约束的线性二次型问题，我们可以解出其分析解，这部分内容我们将在 1.3.7 节展开。在另一类常见的最优控制问题中，状态和/或控制则会受到一些线性约束。相应地，在上述例子中，引入由某些给定标量 a_k，b_k，c_k 和 d_k 定义的状态和/或控制的约束 $a_k \leqslant x_k \leqslant b_k$ 和 $c_k \leqslant u_k \leqslant d_k$ 也会是一个很自然的选择。这种存在约束的问题不仅可以采用动态规划求解，而且也可以采用二次规划的方法解决。一般来说，具有连续的状态及控制空间的、确定性最优控制问题，（除了可以用动态规划解决外）可以通过非线性规划的方式求解，例如梯度法、共轭梯度法和牛顿法。根据所求问题的特殊结构，我们还可以对上述求解方法做相应的调整。

1.1.2 动态规划算法

在本节中，我们将介绍动态规划算法并证明其合理性。这种算法基于一个简单的概念，即最优性原理（*principle of optimality*）。现将该原理概述如下；见图1.1.5。

最优性原理. 给定初始状态 x_0，记相应的某一最优控制序列为 $\{u_0^*, \ldots, u_{N-1}^*\}$。在该控制序列作用于状态方程(1.1)后，系统的状态序列为 $\{x_1^*, \ldots, x_N^*\}$。先考虑如下子问题：从 k 时刻的状态 x_k^* 出发，以 $\{u_k, \ldots, u_{N-1}\}$，$u_m \in U_m(x_m)$，$m = k, \ldots, N-1$ 为优化变量，我们希望最小化从 k 时刻到 N 时刻累积的“展望费用”

$$g_k(x_k^*, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N).$$

那么，通过截短原问题的最优控制序列而得到的 $\{u_k^*, \ldots, u_{N-1}^*\}$ 就是该子问题的最优解。

上述的子问题被称为始于 x_k^* 的尾部子问题（*tail subproblem*）。简单来讲，最优性原理就是指出一个最优控制序列的尾部即为其相应的尾部子问题的最优解（*the tail of an optimal sequence is optimal for the tail subproblem*）。

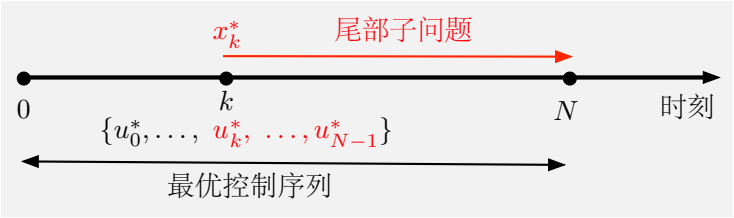

图 1.1.5: 最优性原理图示。对于一个最优控制序列 $\{u_0^*, \ldots, u_{N-1}^*\}$，其尾部 $\{u_k^*, \ldots, u_{N-1}^*\}$ 是一个新的最优控制问题的最优控制序列。我们记原序列相应的最优轨迹为 $\{x_1^*, \ldots, x_N^*\}$，那么这个新的问题就是以该最优轨迹中的状态 x_k^* 为初始状态的原问题的尾部子问题。

从直观上讲，最优性原理的论证很容易。假设截取的控制序列 $\{u_k^*, \ldots, u_{N-1}^*\}$ 不是相应子问题的最优解。那么在求解原问题的过程中，当到达状态 x_k^* 时，通过将剩余的控制序列替换为以上假设的子问题的最优解（因为到达状态 x_k^* 前采用的控制序列 $u_0^*, \ldots, u_{k-1}^*$ 并不会约束后续的控制选择，所以这么做是可行的），我们将得到比原最优解对应的原问题的最低费用更低的总费用。如果以自驾出行来类比，假设从洛杉矶到波士顿的最快路径途经芝加哥。那么最优性原理就是如下显然的事实：以上最快路径中从芝加哥到波士顿的部分也是从芝加哥开始到波士顿结束的最快路径。

最优性原理表明我们可以从后往前地、逐步地求解最优费用函数：首先针对仅涉及最后一段的“尾部子问题”求解其最优费用函数，然后求解涉及最后两个阶段的“尾部子问题”，依此类推，直到求得整个问题的最优费用函数。

动态规划算法就是基于上述观念：该算法依序执行，通过利用已知的更短时间跨度的尾部子问题的解，解决某一给定的时间跨度的所有尾部子问题（*solving all the tail subproblems of a given time length, using the solution of the tail subproblems of shorter time length*）。我们以例1.1.1中介绍的调度问题来说明上述算法。相关的计算过程简单而乏味，且跳过这些计算步骤也不影响后续内容的理解。然而，对于动态规划算法的初学者，通读这些计算步骤仍然是有益的。

例 1.1.1 (一个确定性调度问题——续)**.** 现在我们考虑例*1.1.1*中介绍的调度问题，并利用最优性原理求解。我们需要以最优的方式来安排工序 A、B、C 和 D。在两个工序交接时会产生一定的转移费用，其相应数值标注在图*1.1.6*中对应弧的一旁。

根据最优性原理可知，原问题的一个最优调度序列的“尾部”必然是相应的尾部子问题的最优解。例如，假设最优调度为 CABD。那么在将前两道工序 C 和 A 确定后剩余两道工序的安排应当是先 B 后 D，即 BD，而不是 DB。考虑到这一点，我们先求解所有长度为二的尾部子问题，接着再求解长度为三的尾部子问题，最后求解长度为四的原问题（对于长度为一的子问题，因为已经安排了三道工序，最后一个决策即为仅有的剩余工序，显然无需求解）。不难看出，一旦长度为 k 的子问题得到解决，那么长度为 $k+1$ 的问题将很容易求解，这就是动态规划算法的核心。

长度为 2 的尾部子问题：这些子问题涉及到两个没有安排的工序，并且对应于状态 AB、AC、CA 和 CD（见图*1.1.6*）。

状态 AB：从该状态出发，接下来唯一可安排的工序为 C，因此该子问题的最优费用为 9（在工序 B 后安排 C 花费 3，二中 C 后安排 D 花费 6，两费用相加可得）。

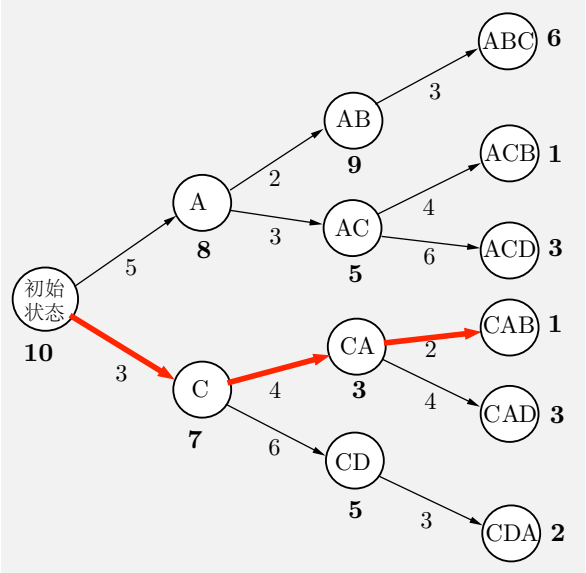

图 1.1.6: 确定性调度问题图示。每个决策的费用标注在相应弧的一旁。每个节点/状态一旁标注的数值代表从该状态为起点进行工序安排的最优解所需的费用。该费用即为相应尾部子问题的最优费用（参照最优性原理）。原问题的最优费用为 10，并标注在初始状态一旁。最优调度由加粗的弧标出。

状态 AC：此处有两种调度方案（a）先安排 B 再安排 D，共计花费 5，或者（b）先安排 D 再安排 B，共计花费 9。两选项中前者更优，因此该尾部子问题的最优费用为 5，并注于图*1.1.6*中节点 AC 旁边。

状态 CA：此处有两种调度方案（a）先安排 B 再安排 D，共计花费 3，或者（b）先安排 D 再安排 B，共计花费 7。两选项中前者更优，因此该尾部子问题的最优费用为 3，并注于图*1.1.6*中节点 CA 旁边。

状态 CD：从该状态出发，接下来唯一可安排的工序为 A，因此该问题的最优费用为 5。

长度为 3 的尾部子问题：现在我们可以通过已知的长度为 *2* 的子问题的最优解来求解这些子问题。

状态 A：从该状态出发可选的调度安排有（a）B（花费 2）并从所得状态为起点求解相应的长度为 2 的子问题的最优解（其最优费用为 9，如前所述），因而总计花费 11，或者（b）先安排 C（花费 3）并从所得状态为起点求解相应的长度为 2 的子问题的最优解（其最优费用为 5，如前所述），因而总计花费 8。上述的第二种调度方案是最优的，其相应的费用 8 即为该尾部子问题的最优费用，因而标注于图*1.1.6*中节点 A 旁边。

状态 C：从该状态出发可选的调度安排有（a）A（花费 4）并从所得状态为起点求解相应的长度为 2 的子问题的最优解（其最优费用为 3，见前述计算），因而总计花费 7，或者（b）先安排 D（花费 6）并从所得状态为起点求解相应的长度为 2 的子问题的最优解（其最优费用为 5，见前述计算），因而总计花费 11。上述的第一种调度方案

是最优的，其相应的费用 7 即为该尾部子问题的最优费用，因而标注于图*1.1.6*中节点 C 旁边。

长度为 4 的原问题：从初始状态出发可选的调度安排有（a）A（花费 5）并从所得状态为起点求解相应的长度为 3 的子问题的最优解（其最优费用为 8，如前所述），因而总计花费 13，或者（b）先安排 C（花费 3）并从所得状态为起点求解相应的长度为 3 的子问题的最优解（其最优费用为 7，如前所述），因而总计花费 10。上述的第二种调度方案是最优的，其相应的费用 10 即为原问题的最优费用，因而标注于图*1.1.6*中初始状态节点旁边。

在利用所有尾部子问题的解计算出原问题的最优费用后，我们可以求得最优调度：我们从初始状态出发，从前向后计算，每一时刻选择最优的调度，即选择从当前状态出发的相应尾部子问题的最优调度的第一步安排。通过这种方式，从图*1.1.6*标注可知，CABD 为最优调度。

通过动态规划求解最优控制序列

在上述章节中，我们给出了一些佐证最优性原理成立的经验性的论据。现在，我们将通过把以上的论证转化成数学术语，从而给出求解确定性有限阶段问题的动态规划算法。该算法按照顺序，从 J_N^* 出发，从后往前求解 J_{N-1}^*，J_{N-1}^* 等，从而依次构造如下函数

$$J_N^*,\, J_{N-1}^*, \ldots, J_0^*.$$

针对确定性有限阶段问题的动态规划算法. 首先令以下方程成立

$$J_N^*(x_N) = g_N(x_N), \quad \text{对所有} x_N, \tag{1.3}$$

对 $k = 0, \ldots, N-1$，令

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big) \Big], \quad \text{对所有} x_k. \tag{1.4}$$

注意到在阶段 k，动态规划算法要求对所有的 x_k 带入式(1.4)计算后，才能进行下一步 $k-1$ 阶段的计算。关于动态规划算法的关键是，对于每个初始状态 x_0，上述算法最后一步得到的数值 $J_0^*(x_0)$ 等于最优费用 $J^*(x_0)$。而且，我们可以证明有如下更一般的结论成立，即对于所有的 $k = 0, 1, \ldots, N-1$，以及 k 时刻的所有状态 x_k，如下等式成立

$$J_k^*(x_0) = \min_{\substack{u_m \in U(x_m) \\ m=k,\ldots,N-1}} J(x_k; u_k, \ldots, u_{N-1}), \tag{1.5}$$

其中

$$J(x_k; u_k, \ldots, u_{N-1}) = g_N(x_N) + \sum_{m=k}^{N-1} g_m(x_m, u_m), \tag{1.6}$$

即 $J_k^*(x_k)$ 是始于时刻 k 的状态 x_k 而终于时刻 N 的 $(N-k)$ 阶段尾部子问题的最优费用。[3]基于这一事实，我们称 $J_k^*(x_k)$ 为在状态 x_k 与时刻 k 的最优展望费用（*optimal cost-to-go*），并称 J_k^* 为时刻 k 的最优展望费用函数（*optimal cost-to-go function*）或者最优费用函数（*optimal cost function*）。在求解最大值的问题中，上述动态规划算法式(1.4)中的最小化运算被最大化运算所取代，而 J_k^* 也被称为 k 时刻的最优价值函数。

[3]我们可以通过归纳法证明上述结论。鉴于 $J_N^*(x_N) = g_N(x_N)$，上述结论在 $k = N$ 时成立。为证明对所有的 k 都成立，

注意到上述算法会求解每个尾部子问题，即计算从所有中间状态出发直到最终阶段所累积的阶段费用，再从中找出对应于所有中间阶段的最小费用。一旦取得函数 $J_0^*,\dots,J_N^*$ 后，对给定的初始状态 x_0，我们可以采用以下前向的算法构造最优控制序列 $\{u_0^*,\dots,u_{N-1}^*\}$ 及其相应的状态轨迹 $\{x_1^*,\dots,x_N^*\}$。

构造最优控制序列 $\{u_0^*,\dots,u_{N-1}^*\}$. 首先令

$$u_0^* \in \arg\min_{u_0\in U_0(x_0)} \Big[g_0(x_0,u_0)+J_1^*\big(f_0(x_0,u_0)\big)\Big],$$

及

$$x_1^* = f_0(x_0,u_0^*).$$

依次向后，对 $k=1,2,\dots,N-1$，令

$$u_k^* \in \arg\min_{u_k\in U_k(x_k)} \Big[g_k(x_k,u_k)+J_{k+1}^*\big(f_k(x_k,u_k)\big)\Big], \tag{1.7}$$

及

$$x_{k+1}^* = f_k(x_k^*,u_k^*).$$

同样的算法也可以用于构造任何尾部子问题的最优控制序列。图1.1.6描绘出了动态规划算法用于规划问题例1.1.1的相关计算。节点旁边的数字给出了相应的展望费用的取值，而加粗的弧则标出了根据上述算法构造的最优控制序列。

1.1.3 值空间的近似

只有当我们通过动态规划算法求出所有 x_k 和 k 对应的 $J_k^*(x_k)$ 的取值后，我们才可以使用前述的前向的算法来构造最优控制序列。然而在实际中，因为可能需要计算的 x_k 和 k 的数量过大，采用动态规划求解时通常会非常耗时。但是，如果以一些近似函数 $\tilde{J}_k$ 代替最优展望费用函数 J_k^*，那么我们就可以采用一个类似的前向求解过程。这就是值空间的近似（*approximation in value space*）的基础，也是本书讲解的核心。在动态规划求解的式(1.7)中，通过以 $\tilde{J}_k$ 来代替 J_k^*，我们就可以相应地求得次优解 $\{\tilde{u}_0,\dots,\tilde{u}_{N-1}\}$，用来代替最优的 $\{u_0^*,\dots,u_{N-1}^*\}$。

值空间的近似——用 $\tilde{J}_k$ 代替 J_k^*. 首先令

$$\tilde{u}_0 \in \arg\min_{u_0\in U_0(x_0)} \Big[g_0(x_0,u_0)+\tilde{J}_1\big(f_0(x_0,u_0)\big)\Big],$$

我们据式(1.5)和(1.6)得到如下关系

$$\begin{aligned}
J_k^*(x_k) &= \min_{\substack{u_m\in U(x_m)\\ m=k,\dots,N-1}} \Big[g_N(x_N)+\sum_{m=k}^{N-1} g_m(x_m,u_m)\Big]\\
&= \min_{u_k\in U_k}\Big[g_k(x_k,u_k)+\min_{\substack{u_m\in U(x_m)\\ m=k+1,\dots,N-1}}\Big[g_N(x_N)+\sum_{m=k+1}^{N-1} g_m(x_m,u_m)\Big]\Big]\\
&= \min_{u_k\in U_k}\Big[g_k(x_k,u_k)+J_{k+1}^*\big(f_k(x_k,u_k)\big)\Big],
\end{aligned}$$

其中最后一个等式是基于归纳假设而成立。此处有一个数学上很微妙的细节。当求最小值的时候，在某些状态 x_k，展望费用函数 J_k^* 可能取值 $-\infty$。即便如此，上述归纳法中的论述依然成立。

及

$$\tilde{x}_1 = f_0(x_0, \tilde{u}_0).$$

依次向后，对 $k = 1, 2, \ldots, N-1$，令

$$\tilde{u}_k \in \arg\min_{u_k \in U_k(\tilde{x}_k)} \Big[g_k(\tilde{x}_k, u_k) + \tilde{J}_{k+1}\big(f_k(\tilde{x}_k, u_k)\big) \Big], \tag{1.8}$$

及

$$\tilde{x}_{k+1} = f_k(\tilde{x}_k, \tilde{u}_k).$$

构造合适的近似展望费用函数 $\tilde{J}_k$ 是强化学习方法的主要关注点之一。针对不同问题的特点，有多种不同的方法可以选择。对这些方法的讲解将会在从第 2 章起的后续章节中依次展开。

Q-因子和 Q-学习

以下表达式

$$\tilde{Q}_k(x_k, u_k) = g_k(x_k, u_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k)\big),$$

即出现在式(1.8)右侧的部分，被称为关于 (x_k, u_k) 的（近似）Q-因子。[4]特别地，近似最优控制(1.8)可通过最小化 Q-因子来得到

$$\tilde{u}_k \in \arg\min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k, u_k).$$

上述分析意味着在值空间近似的方案中采用 Q-因子代替费用函数成为可能。这种类型的方法以一种替代的（并且等价的）形式的动态规划算法作为出发点。这种替代形式的算法并不生成最优展望费用函数 J_k^*，取而代之的是对于所有的状态-控制对 (x_k, u_k) 和时刻 k 生成的如下最优 Q-因子（*optimal Q-factors*）

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big). \tag{1.9}$$

因此最优 Q-因子即是动态规划等式(1.4)的右侧被最小化的表达式。我们注意到该等式表明最优费用函数 J_k^* 可以由最优 Q-因子 Q_k^* 通过以下计算获得

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} Q_k^*(x_k, u_k).$$

此外，通过上述等式，动态规划算法可以被重写作一种本质上等价的、仅涉及 Q-因子的形式

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + \min_{u_{k+1} \in U_{k+1}\left(f_k(x_k, u_k)\right)} Q_{k+1}^*\big(f_k(x_k, u_k), u_{k+1}\big).$$

在后续讲解被称作Q-学习（*Q-learning*）的这一类强化学习方法时，我们会介绍相关算法的精确与近似形式。

---

[4] “Q-学习”一词及一些相关的算法思想是 Watkins 在论文 [Wat89] 中首次提出的（Watkins 以符号 Q 代表 Q-因子，这一算法因该符号而得名）。“Q-因子”一词被用在专著 [BT96] 中，并被用于本书中。Watkins 在 [Wat89] 中则称之为（在一个给定状态的）“动作价值”。“状态-动作价值”和“Q-价值”这些名称在文献中也很常见。

参考文献

[Ber91] Dimitri P. Bertsekas. *Linear network optimization: algorithms and codes*. Mit Press, 1991.

[Ber98] Dimitri P. Bertsekas. *Network optimization: continuous and discrete models*. Athena Scientific Belmont, 1998.

[Ber17] Dimitri P. Bertsekas. *Dynamic programming and optimal control: Vol. I*. Athena scientific Belmont, 4th edition, 2017.

[BT96] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[Wat89] Christopher J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, 1989.