# 动态规划算法在管理会计中的应用

钟方源

【摘要】动态规划算法是运用状态转移解决多阶段决策的一种最优化方法。这种方法基于最优化原理，把多阶段过程转化为一系列单阶段问题，进而逐个求解，可以高效地解决许多用贪心算法或分治算法无法解决的问题。本文结合管理会计中的问题，运用运筹学、金融建模等知识，探讨了动态规划算法在管理会计中的应用。

【关键词】动态规划；管理会计；最优化原理

【中图分类号】F230　　【文献标识码】A　　【文章编号】1004-0994(2016)05-0053-3

## 一、应用动态规划算法的动机

案例1：一家公司现有m万元，本年可投资的业务有n项。其中，第一项业务对应的成本是$C_1$，收益是$R_1$；第二项业务对应的成本是$C_2$，收益是$R_2$……第n项业务对应的成本是$C_n$，收益是$R_n$。假设同种业务只能投资一次，不可重复。若每项业务对应的成本和收益已知，求该公司用m万元投资这些业务可以取得的最大收益。

显然可以用递归法来解决此问题：

设V(i,j)表示能够用j万元购买的前i项业务收益最大的子集的收益。根据V(i,j)这个最佳子集中是否包含业务i，可以得到下列递归关系式：

$$V(i,j)=\begin{cases}\max\{V(i-1,j),R_i+V(i-1,j-C_i)\}, & j-C_i\geqslant 0\\ V(i-1,j), & j-C_i<0\end{cases}$$

该方程表示的是：如果第i项业务的成本$C_i$比j小(或等于j)，那么取得最大收益的方案有可能包含该项业务，至于是否包含，就看包含该业务所能取得的最大收益与不含该业务所能取得的最大收益两者中何者更大。而如果第i项业务的成本$C_i$比j大，那就不能选择购买该项业务，也就是说取得最大收益的方案一定不包含此项业务。

当i和j有至少一项等于零时，表示该公司投资业务的资金为零或者不投资任何业务，所以其所能取得的收益一定为零，即当i=0或j=0时，V(i,j)=0，故该方程的边界条件是：

$V(0,j)=0, j\geqslant 0$

$V(i,0)=0, i\geqslant 0$

该案例目标是求出V(n,m)，即用m万元购买的n项业务收益最大的子集的收益。

因此，可以得到以下递归过程(伪代码)：

```
function V(i,j:integer);
begin
  if (i=0) or (j=0) then return 0
  else begin
if j-Ci>=0 then
  Answer:=max[V(i-1,j),Ri+V(i-1,j-Ci)];
if j-Ci<0 then
  Answer:=V(i-1,j);
end;
end;
```

**1.** 记忆化搜索。上面的递归算法显然是正确的，但是它的运算速度却很慢，因为它的时间复杂度是指数级的。如果记V(i,j)的值是d[i,j]，以数组来表示，由于1≤i≤n，1≤j≤m，所以一共只有O(n×m)个d值需要计算，而在执行递归算法的时候却做了大量的重复运算。

可以这样改进这个算法：

因为V(i,j)的值一旦被计算出来就不会改变，所以在每次调用V函数之前先检查之前是否已经计算过该值，如果是，则直接从数组中读出，不必再花费时间重新进行计算，即：

```
function V(i,j:integer);
begin
  if Calculated[i,j] then return d[i,j];
  //此处为原来的V函数代码
  d[i,j]:=Answer;
  Calculated[i,j]:=true;
end;
```

**2.** 自底向上的递推。除了上述这种改进方法，还有另外一种改进方法：可以按照一定的顺序计算所有的d值。由于计算d[i,j]需要知道d[i-1,j]和d[i-1,j-$C_i$]，所以可以按照i~j递增的顺序来计算d[i,j]，即：

```
for i=0 to n do d[i,0]=0;
for j=0 to m do d[0,j]=0;
for i:=1 to n do
for j:=1 to m do
  begin
if (j-Ci<0) then
  d[i,j]:=d[i-1,j];
    else
      d[i,j]:=max(d[i-1,j],Ri+d[i-1,j-Ci]);
  end;
```

这两种改进方法就是动态规划算法。正是由于递归算法中产生了大量的重叠子问题，导致运算速度很慢，因此才有动态规划算法所带来的两种改进。

d值的递推式被称为状态转移方程，每个d值则被称为一种状态。在下文中，因篇幅所限，不再写出伪代码，而只把d值的递推式以及边界值写出来。因为只要有了递推式和边界值，就很容易将伪代码写出来，不论是记忆化搜索还是自底向上的递推。一般而言，伪代码是一个三重循环：第一重循环按照一定的逻辑顺序循环每一个状态；第二重循环则考虑不同的递推路径，这些递推路径的数量被称为决策数；最里层循环进行单个状态的状态转移。因此，动态规划算法的复杂度是状态数、决策数与转移费用的乘积。

**二、动态规划算法的适用条件**

**1.** *最优化原理*。在案例1中运用到这个问题的一个性质，在把原问题转化成子问题时，当且仅当子问题最优时，原问题最优，这就是最优化原理。最优化原理并不是显而易见的，也不是一定成立的。比如在案例1中，如果改成求个位数字最大的收益，则动态规划算法失效。如果最优化原理成立，则称该问题有最优子结构。

**2.** *无后效性*。无后效性是指每一种状态的值只取决于当前状态所对应的特征变量，而与到达该状态的运算方式无关。如果不满足无后效性，那么状态转移方程的表达就是不合理的，因为同一个状态可能会对应很多本质不同的实例。在案例1中，对于给定的资金总数m、业务数量n，以及每个业务对应的成本$C_i$和收益$R_i$而言，一旦计算出某个状态的d值，就不会再发生改变，因此才可以运用记忆化搜索的方法用数组把它记录下来。而如果d值会因后续结点值的变化而发生改变，那么不仅记忆化搜索的实现方式不再适用，而且反方向求解的递推算法也很可能出错。

**3.** *子问题的重叠性*。在案例1中，动态规划算法将原来具有指数级复杂度的递归算法改进成只具有多项式时间复杂度的算法，其关键就在于减少了重复冗余的运算，这就是应用动态规划算法的根本目的。动态规划算法的实质是一种以空间存储量换时间运算速度的算法技术。它在实现的过程中，不得不对运算过程中的各个状态值进行存储，以方便后续的运算调用已经得出的值，从而降低时间复杂度。因此，使用动态规划算法来解决的问题一般具有一个显著特征：子问题的重叠性。如果子问题不是重叠的，那么动态规划算法对每一状态进行存储的这一行为就是不必要的，甚至是浪费空间资源的。虽然子问题的重叠性并不是动态规划算法适用的必要条件，但是只有在子问题具备重叠性的前提下，应用动态规划算法与其他算法相比才具有时间效率上的优势。

**三、管理会计中常见的动态规划模型**

**1.** *背包模型*。背包模型的一般描述是：有n件物品和一个容量为m的背包。在背包中放入第i件物品会消耗数量为$C_i$的费用，而能够得到的对应价值是$R_i$。求在背包中放入哪些物品，可以使这些物品既不会超出背包的容量，又能够使得背包的价值最大化。同种物品或者投资不可重复选择的背包模型被称为01背包。下面再分析另一种背包模型。

案例2：假设一家公司现有m万元，本年可投资的业务有n项。其中，第一项业务对应的成本是$C_1$，收益是$R_1$；第二项业务对应的成本是$C_2$，收益是$R_2$……第n项业务对应的成本是$C_n$，收益是$R_n$。假设对同一业务可以重复投资。若每项业务对应的成本和收益已知，求该公司用m万元投资这些业务可以取得的最大收益。

这种可重复选择同种投资方案的背包模型被称为完全背包。这个问题与案例1十分相似，不同之处仅在于业务可以重复投资。所以对于每一种业务而言，可选的策略不是投资或者不投资两种选择，而是投资0次、1次……直至$m/C_i$次等多种选择。设k为投资的次数，故$0\leqslant k\leqslant m/C_i$，可以由案例1的状态转移方程改进而得到下列方程：

$$d[i,j]=\max\{d[i-1,j],d[i-1,j-k\times C_i]+k\times R_i\},0\leqslant k\leqslant m/C_i$$

此时的边界条件是：

$$d[i,j]=0,j\geqslant 0$$

$$d[i,0]=0,i\geqslant 0$$

**2.** *资源分配模型*。资源分配模型的一般模式是：给定数量为m的资源，用以分配给n个部门，第i个部门获得j数量的资源时有对应的盈利值R[i,j]，求分配m数量的资源给n个部门所能获得的最大盈利。

案例3：某公司有m万元的资金，分配给n个部门。已知第i($1\leqslant i\leqslant n$)个部门获得j($0\leqslant j\leqslant m$)万元的资金时可以为该公司带来R[i,j]万元的盈利。求将这m万元资金分配给n个部门所能获得的最大盈利。

用数组d[i,j]表示前i个部门分配j万元资金所能够带来的最大盈利数，则状态转移方程为：

$$d[i,j]=\max\{d[i,j],d[i-1,k]+R[i,j-k]\}$$

其中，$1\leqslant i\leqslant n$，$0\leqslant j\leqslant m$，$0\leqslant k\leqslant j$。k表示具体分配给前i-1个部门的资金。该方程表示的是，若给前i-1个部门分配k万元资金，并给第i号部门分配j-k万元所得到的盈利更大，那么就用这个方案更新d[i,j]的值。

不把任意数量的资金分配给任何部门所能够得到的盈利数只能是零，所以该方程的边界条件是：

$d[0,j]=0, 0\leqslant j\leqslant m$

**3.** 固定资产投资模型。大型固定资产的投资一般分期进行，在每一会计期间又有多种投资方案可供选择，每个决策都会对总体效益产生影响，因此可以用动态规划算法来计算最优的投资策略。

案例4：某公司拟在n年内建设年产量相同的m个生产工厂以适应发展。根据市场对产品的需求量，该公司预测各年累计需求的工厂数为$X_1, X_2, \cdots, X_n$，各年建设一间生产工厂所需的成本分别为$C_1, C_2, \cdots, C_n$。建厂期为一年，可以当年建成。假设不考虑货币的时间价值及贴现问题，求符合生产需要的最少投资成本。

用数组d[i,j]表示第i($0\leqslant i\leqslant n$)年建设j($0\leqslant j\leqslant m$)家工厂所需要的最少投资成本。设k($0\leqslant k\leqslant j$)为循环变量，循环每一个可建工厂数的值。则状态转移方程是：

$d[i,j]=\min\{d[i-1,j], d[i-1,k]+(j-k)\times C_i, j\geqslant X_i\}$

这个方程表示的是第i年建设j家工厂所需要的最少投资成本等于第(i-1)年建设k家工厂的最少投资成本加上第i年建设(j-k)家工厂的投资成本。在该方程中，k从0循环到j，尝试每一种方案，取投资费用最小的方案的值。

不消耗任何时间且不建设任何工厂所消耗的成本值是零，故该方程的边界条件是$d[0,0]=0$。

**4.** 生产—存贮模型。该模型的一般模式是：企业根据市场需求量制定今后每个时期的生产批量计划。即根据预测的各时期产品的需求量以及企业的生产能力和贮存能力选择最佳的生产—存贮策略，调整各个时期的生产批量，使得企业的生产既能够满足市场的需求，又可以尽量减少损失。在市场需求量小时，适当多生产一些商品留至库存，以弥补市场需求量大时生产能力之不足。其总策略目标是追求整个计划期内总成本最低。

案例5：已知某公司在n个月内根据市场需求生产的机器产品数量分别为$X_1, X_2, \cdots, X_n$。若该公司的生产能力为每月60部，库存能力为40部，每批生产20部。每部机器生产费为2万元，当月生产的机器若不能售出，每一部机器需要付出的存贮费是5万元。求最优生产—存贮策略所花费的最低总成本。

用数组d[i,j]表示前i个月共生产了j批次，设k是循环变量，循环尝试上一期(即i-1月)共生产的批次数。得到状态转移方程：

$$d[i,j]=\min\{d[i,j], d[i-1,k]+(k\times 20-\sum_{1}^{i-1}X_n)\times 5+(j-k)\times 20\times 2\}$$

其中：$(k\times 20-\sum_{1}^{i-1}X_n)$表示当期库存数量；$(j-k)\times 20$表示的是本期生产的台数。该方程的边界条件是：

$d[1,j]=j\times 20\times 2$

这是由于在第一个月无须考虑上一期库存，只需计算出第一个月的产品生产费用。

**四、管理会计中常见的动态规划优化方法**

**1.** 问题的稀疏性。有时候，动态规划面对的问题是稀疏的，即可优化为只需计算一部分子问题的解。此时需要把算法优化至必须计算的子集的规模。对于非常稀疏的问题，动态规划优化方法改进的效果是十分可观的。一般方法是先通过预处理计算出必须计算的子集，从而排除那些不需要计算的子集。

比如案例2中，如果业务A和业务B对应的成本和收益满足：$C_A\leqslant C_B$且$R_A\geqslant R_B$，则可以直接把业务B去掉，不再考虑。这是由于业务A的成本比业务B的成本小，但是业务A的收益比业务B的收益大，即业务A比业务B更“物美价廉”。在同一业务可以无限次重复投资的情况下，总是会优先选择A而不选择B，所以业务B可以直接被排除出必须计算的子集。但是在案例1中，由于同一业务不可重复投资，故仍然有可能选择业务B，此优化方法不再适用。

**2.** 状态压缩。运用动态规划算法解决问题最关键的步骤就是得出状态转移方程的表达式。一般而言，一个数组即可表示出所有状态。但是经济生活中也会遇到一些问题，这类问题既具有动态规划算法的适用性质，又包含大量需要表示的信息。表示这些问题的状态如果用数组进行存储的话，可能需要运用到高维。但是由于动态规划算法的特点是用空间效率换取时间效率，高维的数组很可能不便于存储。于是，需要通过状态压缩的方法来降低维数存储状态。

比如在案例1中，状态转移方程为：

$d[i,j]=\max\{d[i-1,j], R_i+d[i-1,j-C_i]\}$

d[i,j]的取值有两种，一种是d[i-1,j]，另一种是$d[i-1,j-C_i]+R_i$。显然，如果d[i,j]=d[i-1,j]的话，相当于d[i,j]直接复制了i-1时d的取值，而j的取值不发生改变。而如果$d[i,j]=d[i-1,j-C_i]+R_i$，注意到i依然对应着i-1，j对应着$j-C_i$，然后再加上$R_i$。也就是每次用到的值只有j、$j-C_i$和$R_i$。因此可以得到压缩后的一维状态转移方程：

$d[j]=\max\{d[j], d[j-C_i]+R_i\}$

其中，d[j]表示的是j万元资金用来投资可以取得的最大收益。

---

**主要参考文献：**

刘汝佳，黄亮.算法艺术与信息学竞赛[M].北京：清华大学出版社，2003.

左志坚.动态规划在管理会计中的应用[J].西北大学学报，1994(2).

**作者单位：**韶关学院经济管理学院，广东韶关 512000