文章编号:1000-5862(2017)02-0116-06

# 3 个变形背包问题的形式化推导

游 颖,杨庆红*,齐蕾蕾
(江西师范大学计算机信息工程学院,江西 南昌 330022)

**摘要:** 在对 0-1 背包问题的若干变形问题进行深入研究的基础上,使用二进制数组的方式形式化描述了几种背包问题的程序规约,通过程序规约变换技术获取问题求解的递推关系,给出了 3 个变形背包问题的算法推导过程,有效保证了算法程序的可靠性,并可将采用的推导方法在子集和问题、船装载等问题中加以推广应用.

**关键词:** 形式化推导; 0-1 背包问题; 二进制向量; 程序规约; 递推关系
**中图分类号:** TP 311.5 **文献标志码:** A **DOI:** 10.16357/j.cnki.issn1000-5862.2017.02.02

## 0 引言

0-1 背包问题可描述为:给定 $n$ 个物品和一个背包,物品 $p_i$ 的质量为 $w_i$,价值为 $v_i$,背包所能容纳的物品的质量为 $C$,要从这 $n$ 个物品中选择若干物品放入背包中,使得放入的物品总质量不超过 $C$,而物品的总价值达到最大[1].

0-1 背包问题在实际应用中存在多种变形问题,如在不考虑价值的情况下恰好将背包装满(可推广到子集和问题[2])、背包有额外的体积容量限制[3]、背包恰好装满的前提下物品数量最少以及双背包问题[4]等. 其中物品数量最少的背包问题可应用到最简换零问题、装载问题上,额外限制背包体积问题可代表一类对背包有多个条件限制的问题,双背包问题可类推到多背包问题,因此本文着重对这 3 个变形背包问题进行形式化推导求解.

关于背包问题的各种算法的求解方法比较多,如动态规划算法、分支限界算法等[5],但这些方法在求解时只是通过人工分析直接给出最终算法,没有给出算法推导过程以及可信的证明过程,因而没有较好解决算法"从何而来"的问题,同时无法保证最终算法的正确性. 基于递推关系的算法形式化推导方法从问题形式化功能规约出发,完成问题分划及规约的数学等价变换[6-9],获得问题求解的递推关系,进而得到该问题的算法程序. 该方法通过确保推导过程的正确性,有效保证了最终所得到的算法程序的正确性. 薛锦云教授及其团队在此方面开展了大量研究工作,定义了规约描述语言和算法描述语言,开发了相关支撑平台和工具,为软件形式化开发提供了有效途径[10-11].

本文将采用基于递推关系的算法形式化推导方法,通过程序规约变换技术[6-7]对 3 个典型变形背包问题进行形式化推导,清晰展示算法的求解过程;由于推导过程是以数学变换为基础的,因而在得到最终算法的同时也有效保证了算法程序的正确性. 本文采用的方法可以进一步推广到其他 0-1 背包问题以及子集和问题、船装载等问题的求解.

## 1 程序规约变换技术预备知识

基于递推关系的算法形式化推导方法通常分为以下 4 步[10-12]:(i) 形式化描述求解问题的程序规约;(ii) 通过程序规约的变换技术得到问题求解的递推关系;(iii) 对递推关系中出现的函数和变量初始化,结合所得到的递推关系推导出问题的求解算法;(iv) 将算法转换成可执行程序.

本文的推导使用了以下量词的性质[13-15]:

1) **范围分裂**,$Q(i:r(i):f(i)) = Q(i:r(i) \wedge b(i):f(i))\ qQ(i:r(i) \wedge \neg b(i):f(i))$;

2) **单一范围**,$Q(i:i = k:f(i)) = f(k)$;

3) **一般分配律**,$Q(i:r(i):gqf(i)) = gq\ Q(i:r(i):f(i))$.

**收稿日期:** 2016-10-22
**基金项目:** 国家自然科学基金(61363013)资助项目.
**通信作者:** 杨庆红(1968-),女,江西南昌人,教授,主要从事软件形式化和智能教育软件的研究. E-mail: yangqh120@163.com

## 2　形式化推导实例

### 2.1　物品数量最少的 0-1 背包问题

问题描述: 给定 $n$ 个物品和一个背包,物品 $p_i$ 的质量为 $w_i$,背包所能容纳物品的质量为 $C$,要从这 $n$ 个物品中选择若干物品放入背包中,使得放入的物品恰好等于背包容量的同时数量最少.

令 $m(i,s)$ 表示从物品 $i$ 开始到物品 $n$,选择若干物品放入容量 $s$ 的背包中,物品总质量恰好等于 $s$ 的最少物品数量. 若该问题无解,则 $m(i,s)=+\infty$

1) 形式化描述求解问题的程序规约.

AQ: 给定 $n$ 种物品 $p_i(1\leqslant i\leqslant n)$,$w[\,]$ 为物品 $p_i$ 的质量,$C$ 为背包的最大容量.

AR: $m(1,C)=(\text{MIN } x:(\Sigma k:1\leqslant k\leqslant n:x[k]*w[k]=C):(\Sigma k:1\leqslant k\leqslant n:x[k]))$,

其中 $x[k]=1$ 表示物品 $p_k$ 可以放入背包,$x[k]=0$ 表示物品 $p_k$ 不可以放入背包.

2) 求解递推关系.

$m(i,s)=(\text{MIN } x:(\Sigma k:i\leqslant k\leqslant n:x[k]*w[k]=s):(\Sigma k:i\leqslant k\leqslant n:x[k]))=(\text{MIN } x:((\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])+(\Sigma k:k=i:x[k]*w[k])=s):(\Sigma k:i+1\leqslant k\leqslant n:x[k])+(\Sigma k:k=i:x[k]))$ {范围分裂} $=(\text{MIN } x:((\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])=s-x[i]*w[i]):(\Sigma k:i+1\leqslant k\leqslant n:x[k])+x[i])$ {单一范围}.

因为 $x[i]$ 的取值只能是 0 或 1,而 $(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])\geqslant 0$,因此有

(i) 当 $s<w[i]$ 时,$x[i]$ 必须取 0,则有

$m(i,s)=(\text{MIN } x:((\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])=s):(\Sigma k:i+1\leqslant k\leqslant n:x[k]))=m(i+1,s)$.

(ii) 当 $s\geqslant w[i]$ 时,$x[i]$ 的值可以取 0,也可以取 1,则有

a) 当 $x[i]=1$ 时,有

$m(i,s)=(\text{MIN } x:((\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])=s-w[i]):(\Sigma k:i+1\leqslant k\leqslant n:x[k]))+1)=(\text{MIN } x:((\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])=s-w[i]):(\Sigma k:i+1\leqslant k\leqslant n:x[k]))+1$ {一般分配律} $=m(i+1,s-w[i])+1$;

b) 当 $x[i]=0$ 时,根据(i) 可以得知 $m(i,s)=m(i+1,s)$. 根据以上分析,当 $s\geqslant w[i]$ 时 $m(i,s)=\min((m(i+1,s-w[i]))+1),m(i+1,s))$.

根据(i) 和(ii) 的推导,可得如下递推关系:

$$m(i,s)=\begin{cases}m(i+1,s), & s<w[i],\\ \min(m(i+1,s-w[i]))+1,\\ \quad m(i+1,s)), & s\geqslant w[i].\end{cases}$$

3) 对递推关系中出现的函数和变量初始化,结合所得到的递推关系推导出问题的求解算法. 具体过程为

Begin: $i=n$;

$$m_{(0<s\leqslant C)}(n,s)=\begin{cases}1, & s=w[n],\\ +\infty, & s\neq w[n],\end{cases}\qquad(1)$$

Termination: $i=0$

Recur:

$$m_{(0<s\leqslant C)}(i,s)=\begin{cases}m(i+1,s), & s<w[i],\\ \min(m(i+1,s-w[i]))+1, & (2)\\ \quad m(i+1,s)), & s\geqslant w[i],\end{cases}$$

End.

4) 将算法转换成可执行程序.

通过以上算法推导,定义数组 $m[1:n,1:C]$,并使用 $m[i,s]$ 去存放 $m(i,s)$ 的值,然后将以上推导关系和算法转换成可执行程序为

```
Begin
s: = 1;   i: = n;
  do  s ≤ C → if  s = w[i] → m[i,s]: = 1;
              ▯ s ≠ w[i] → m[i,s]: = +∞;
              fi
      s: = s + 1;
  od;
  i: = n - 1;
  do i ≥ 1 → s: = 1;
    do s ≤ C →
      if s < w[i] → m[i,s]: = m[i+1,s];
       ▯ s ≥ w[i] → m[i,s]: = min((m[i+1,
       s - w[i]] + 1),m[i+1,s]);
        fi
        s: = s + 1;
    od;
    i: = i - 1;
  od;
End.
```

### 2.2　限制背包体积的 0-1 背包问题

问题描述: 给定 $n$ 个物品和一个背包,物品 $p_i$ 的质量为 $w_i$,体积为 $b_i$,价值为 $v_i$,背包所能容纳物品的质量为 $C$,背包的体积为 $D$,要从这 $n$ 个物品中选择若干物品放入背包中,使得放入的物品质量和体积不超过背包但物品的价值最大.

令 $m(i,j,u)$ 表示从物品 $i$ 开始到物品 $n$,选择若干物品放入容质量为 $j$,体积为 $u$ 的背包所获得的最大价值.

1) 形式化描述求解问题的程序规约.

AQ: 给定 $n$ 种物品 $p_i(1\leqslant i\leqslant n)$,$w[\,]$、$b[\,]$ 和 $v[\,]$ 分别是物品 $p_i$ 的质量、体积和价值,$C$ 是背包的最大容质量,$D$ 是背包的最大体积.

AR: $m(1,C,D)=(\mathrm{MAX}\ x:(\sum k:1\leqslant k\leqslant n:x[k]*w[k])\leqslant C\wedge(\sum k:1\leqslant k\leqslant n:x[k]*b[k])\leqslant D:(\sum k:1\leqslant k\leqslant n:x[k]*v[k]))$,

其中 $x$ 是一个 $n$ 元数组,$x[k]=1$ 表示物品 $p_k$ 可放入背包,$x[k]=0$ 表示物品 $p_k$ 不可放入背包.

2) 求解问题的递推关系.

$m(i,j,u)=(\mathrm{MAX}\ x:(\sum k:i\leqslant k\leqslant n:x[k]*w[k])\leqslant j\wedge(\sum k:i\leqslant k\leqslant n:x[k]*b[k])\leqslant u:(\sum k:i\leqslant k\leqslant n:x[k]*v[k]))=(\mathrm{MAX}\ x:(\sum k:i+1\leqslant k\leqslant n:x[k]*w[k])+(\sum k:k=i:x[k]*w[k])\leqslant j\wedge(\sum k:i+1\leqslant k\leqslant n:x[k]*b[k])+(\sum k:k=i:x[k]*b[k])\leqslant u:(\sum k:i+1\leqslant k\leqslant n:x[k]*v[k])+(\sum k:k=i:x[k]*v[k]))$ {范围分裂} $=(\mathrm{MAX}\ x:(\sum k:i+1\leqslant k\leqslant n:x[k]*w[k])+x[i]*w[i]\leqslant j\wedge(\sum k:i+1\leqslant k\leqslant n:x[k]*b[k])+x[i]*b[i]\leqslant u:(\sum k:i+1\leqslant k\leqslant n:x[k]*v[k])+x[i]*v[i])$ {单一范围} $=(\mathrm{MAX}\ x:(\sum k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant j-x[i]*w[i]\wedge(\sum k:i+1\leqslant k\leqslant n:x[k]*b[k])\leqslant u-x[i]*b[i](\sum k:i+1\leqslant k\leqslant n:x[k]*v[k])+x[i]*v[i])$,由于 $x[i]$ 的取值只能是 0 或 1,并且 $(\sum k:i+1\leqslant k\leqslant n:x[k]*w[k])\geqslant 0$,$(\sum k:i+1\leqslant k\leqslant n:x[k]*b[k])\geqslant 0$,因此有

(i) 当 $0\leqslant j<w[i]$ 或 $0\leqslant u<b[i]$ 时,$x[i]$ 取值必须为 0,则有,

$m(i,j,u)=(\mathrm{MAX}\ x:(\sum k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant j\wedge(\sum k:i+1\leqslant k\leqslant n:x[k]*b[k])\leqslant u:(\sum k:i+1\leqslant k\leqslant n:x[k]*v[k]))=m(i+1,j,u)$.

(ii) 当 $j\geqslant w[i]$ 且 $u\geqslant b[i]$ 时,$x[i]$ 可以取 0,也可以取 1,则有

(a) 当 $x[i]=1$ 时,

$m(i,j,u)=(\mathrm{MAX}\ x:(\sum k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant j-w[i]\wedge(\sum k:i+1\leqslant k\leqslant n:x[k]*b[k])\leqslant u-b[i]:(\sum k:i+1\leqslant k\leqslant n:x[k]*v[k])+v[i])=(\mathrm{MAX}\ x:(\sum k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant j-w[i]\wedge(\sum k:i+1\leqslant k\leqslant n:x[k]*b[k])\leqslant u-b[i]:(\sum k:i+1\leqslant k\leqslant n:x[k]*v[k]))+v[i]=m(i+1,j-w[i],u-b[i])+v[i]$.

(b) 当 $x[i]=0$ 时,根据(ii) 可以得知 $m(i,j,u)=m(i+1,j,u)$.

通过以上分析,当 $j\geqslant w[i]$ 且 $u\geqslant b[i]$ 时 $m(i,j,u)=\max(m(i+1,j,u),m(i+1,j-w[i],u-b[i])+v[i])$. 根据(i) 和(ii) 的推导,可以得到如下递推关系

$$m(i,j,u)=\begin{cases}m(i+1,j,u), & 0\leqslant j<w[i]\ \text{或}\ 0\leqslant u<b[i];\\ \max(m(i+1,j,u),m(i+1,j-w[i],u-b[i])+v[i]), & j\geqslant w[i],u\geqslant b[i].\end{cases}$$

3) 对递推关系中出现的函数和变量初始化,结合所得到的递推关系推导出问题的求解算法.

$$\begin{aligned}&\text{Begin: } i=n;\\ &\underset{(0<j\leqslant C,0<u\leqslant D)}{m(n,j,u)}=\\ &\begin{cases}0, & 0\leqslant j<w[n]\ ||\ 0\leqslant u<b[n];\\ v[n], & j\geqslant w[n]\ \&\&\ u\geqslant b[n];\end{cases}\\ &\text{Termination: } i=0\\ &\text{Recur}\\ &\underset{(0<j\leqslant C,0<u\leqslant D)}{m(i,j,u)}=\\ &\begin{cases}m(i+1,j,u), & 0\leqslant j<w[i]\ ||\ 0\leqslant u<b[i];\\ \max(m(i+1,j,u),m(i+1,j-w[i],u-b[i])+v[i]), & j\geqslant w[i]\ \&\&\ u\geqslant b[i];\end{cases}\\ &\text{End.}\end{aligned}\tag{3}$$

4) 将算法转换成可执行程序.

通过以上算法推导,定义数组 $m[1:n,1:C,1:D]$,并使用 $m[i,j,u]$ 去存放 $m(i,j,u)$ 的值,然后将以上推导关系和算法转换成可执行程序为

```
i: = n;  j: = 1;  u: = 1;
do j ≤ C →
    do u ≤ D →
    if  j < w[i] ‖ u < b[i] → m[i,j,u]: = 0;
     ▯  j ≥ w[i] &&u ≥ b[i] →
       m[i,j,u]: = v[i];
    u + +;
    od;
    j + +;
od;
i: = n - 1;
do i ≥ 1 → j: = 1; u: = 1;
    do j ≤ C →
    do u ≤ D →
```

```
    if j < w[i] ∥ u < b[i] → m[i,j,u] := m[i+1,j,u];
    ▯ j ≥ w[i] &&u ≥ b[i] → m[i,j,u] := max(m[i+1,j,u], m[i+1,j-w[i],u-b[i]] + v[i]);
    fi
      u++;
    od;
    j++;
    od
    i := i-1;
od.
End.
```

### 2.3　双背包问题

**问题描述: 给定 $n$ 个物品和 2 个背包,物品 $p_i$ 的质量为 $w_i$,价值为 $v_i$,2 个背包所能容纳物品的质量分别为 $C_1$ 和 $C_2$,要从这 $n$ 个物品中选择若干物品放入 2 个背包中,使得每个背包的物品不超过背包的容量但物品的总价值最大.**

**令 $m(i,s,t)$ 表示从物品 $i$ 开始到物品 $n$,选择若干物品放入容量分别是 $s$ 和 $t$ 的背包所获得的最大价值.**

1) **形式化描述求解问题的程序规约.**

AQ: **给定 $n$ 种物品 $p_i(1\leqslant i\leqslant n)$,$w[i]$ 和 $v[i]$ 分别是物品 $p_i$ 的质量和价值,$C_1$ 和 $C_2$ 分别是 2 个背包的容量.**

AR: $m(1,C_1,C_2) = (\mathrm{MAX}\,x,y:(\Sigma k:1\leqslant k\leqslant n:x[k]*w[k])\leqslant C_1 \wedge (\Sigma k:1\leqslant k\leqslant n:y[k]*w[k])\leqslant C_2:(\Sigma k:1\leqslant k\leqslant n:(x[k]+y[k])*v[k]))$,**其中 $x$ 和 $y$ 都是 $n$ 元数组,$x[k]=1$ 表示物品 $p_k$ 可以放入第 1 个背包,$x[k]=0$ 表示物品 $p_k$ 不可以放入第 1 个背包. $y[k]=1$ 表示物品 $p_k$ 可以放入第 2 个背包,$y[k]=0$ 表示物品 $p_k$ 不可以放入第 2 个背包. $x[k]$ 和 $y[k]$ 不能同时为 1.**

2) **求解问题的递推关系.**

$m(i,s,t) = (\mathrm{MAX}\,x,y:(\Sigma k:1\leqslant k\leqslant n:x[k]*w[k])\leqslant s\wedge(\Sigma k:1\leqslant k\leqslant n:y[k]*w[k])\leqslant t:(\Sigma k:1\leqslant k\leqslant n:(x[k]+y[k])*v[k])) = (\mathrm{MAX}\,x,y:(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])+(\Sigma k:k=i:x[k]*w[k])\leqslant s\wedge(\Sigma k:i+1\leqslant k\leqslant n:y[k]*w[k])+(\Sigma k:k=i:y[k]*w[k])\leqslant t:(\Sigma k:i+1\leqslant k\leqslant n:(x[k]+y[k])*v[k])+(\Sigma k:k=i:(x[k]+y[k])*v[k])) = (\mathrm{MAX}\,x,y:(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant s-x[i]*w[i]\wedge(\Sigma k:i+1\leqslant k\leqslant n:y[k]*w[k])\leqslant t-y[i]*w[i]:(\Sigma k:i+1\leqslant k\leqslant n:(x[k]+y[k])*v[k])+(x[i]+y[i])*v[i])$.

**因为 $x[i]$ 和 $y[i]$ 的取值只能是 0 或 1,并且 $(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])\geqslant 0$ $(\Sigma k:i+1\leqslant k\leqslant n:y[k]*w[k])\geqslant 0$,因此,有**

(i) **当 $s<w[i]$ 且 $t<w[i]$ 时,$x[i]$ 和 $y[i]$ 必须取 0,则有**

$m(i,s,t) = (\mathrm{MAX}\,x,y:(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant s)\wedge(\Sigma k:i+1\leqslant k\leqslant n:y[k]*w[k])\leqslant t:(\Sigma k:i+1\leqslant k\leqslant n:(x[k]+y[k])*v[k])) = m(i+1,s,t)$.

(ii) **当 $s<w[i]$ 且 $t\geqslant w[i]$ 时,$x[i]$ 必须取 0,$y[i]$ 可以取 0,也可以取 1,则有**

(a) **当 $y[i]=1$ 时,有**

$m(i,s,t) = (\mathrm{MAX}\,x,y:(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant s)\wedge(\Sigma k:i+1\leqslant k\leqslant n:y[k]*w[k])\leqslant t-w[i]:(\Sigma k:i+1\leqslant k\leqslant n:(x[k]+y[k])*v[k])+v[i])) = (\mathrm{MAX}\,x,y:(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant s)\wedge(\Sigma k:i+1\leqslant k\leqslant n:y[k]*w[k])\leqslant t-w[i]:(\Sigma k:i+1\leqslant k\leqslant n:(x[k]+y[k])*v[k]))+v[i] = m(i+1,s,t-w[i])+v[i]$.

(b) **当 $y[i]=0$ 时,根据(i),可以得知 $m(i,s,t) = m(i+1,s,t)$. 通过以上分析,当 $s<w[i]$ 且 $t\geqslant w[i]$ 时有**

$m(i,s,t) = \max(m(i+1,s,t),m(i+1,s,t-w[i])+v[i])$.

(iii) **当 $s\geqslant w[i]$ 且 $t<w[i]$ 时,$y[i]$ 必须取 0,$x[i]$ 可以取 0,也可以取 1,则有**

(a) **当 $x[i]=1$ 时,**

$m(i,s,t) = (\mathrm{MAX}\,x,y:(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant s-w[i])\wedge(\Sigma k:i+1\leqslant k\leqslant n:y[k]*w[k])\leqslant t:(\Sigma k:i+1\leqslant k\leqslant n:(x[k]+y[k])*v[k])+v[i])) = (\mathrm{MAX}\,x,y:(\Sigma k:i+1\leqslant k\leqslant n:x[k]*w[k])\leqslant s-w[i])\wedge(\Sigma k:i+1\leqslant k\leqslant n:y[k]*w[k])\leqslant t:(\Sigma k:i+1\leqslant k\leqslant n:(x[k]+y[k])*v[k]))+v[i] = m(i+1,s-w[i],t)+v[i]$.

(b) 当 $x[i]=0$ 时,根据(i)可以得知:$m(i,s,t)=m(i+1,s,t)$,通过以上分析,当 $s\geqslant w[i]$ 时 $t<w[i]$ 时,有

$$m(i,s,t)=\max(m(i+1,s,t),m(i+1,s-w[i],t)+v[i]).$$

(iv) 当 $s\geqslant w[i]$ 时 $t\geqslant w[i]$ 时,$x[i]$ 和 $y[i]$ 可以取 0,也可以取 1,但 2 者不可同时为 1(因为一个物品不能同时放入 2 个背包),则有

(a) 当 $x[i]=0,y[i]=0$ 时,根据(i)可得 $m(i,s,t)=m(i+1,s,t)$

(b) 当 $x[i]=0,y[i]=1$ 时,根据(ii)可得 $m(i,s,t)=m(i+1,s,t-w[i])+v[i]$.

(c) 当 $x[i]=1y[i]=0$ 时,根据iii)可得 $m(i,s,t)=m(i+1,s-w[i],t)+v[i]$

根据(i) ~ (iv)的推导,可得如下递推关系

$$m(i,s,t)=\begin{cases}m(i+1,s,t), & s<w[i]\ \text{且}\ t<w[i];\\ \max(m(i+1,s,t),m(i+1,s,t-w[i])+v[i]), & s<w[i]\ \text{且}\ t\geqslant w[i];\\ \max(m(i+1,s,t),m(i+1,s-w[i],t)+v[i]), & s\geqslant w[i]\ \text{且}\ t<w[i];\\ \max(m(i+1,s,t),m(i+1,s,t-w[i])+v[i],\\ \quad m(i+1,s-w[i],t)+v[i]), & s\geqslant w[i]\ \text{且}\ t\geqslant w[i].\end{cases}\tag{5}$$

3) 对递推关系中出现的函数和变量初始化,结合所得到的递推关系推导出问题的求解算法为

Begin: $i=n$;

$$m_{(0<s\leqslant C_1,0<t\leqslant C_2)}(n,s,t)=\begin{cases}0, & s<w[n]\ \&\&\ t<w[n];\\ v[n], & s\geqslant w[n]\ \|\ t\geqslant w[n];\end{cases}\tag{6}$$

Termination: $i=1$

Recur

$$m_{(0<s\leqslant C_1,0<t\leqslant C_2)}(i,s,t)=\begin{cases}m(i+1,s,t), & s<w[i]\ \&\&\ t<w[i];\\ \max(m(i+1,s,t),m(i+1,s,t-w[i])+v[i]), & s<w[i]\ \&\&\ t\geqslant w[i];\\ \max(m(i+1,s,t),m(i+1,s-w[i],t)+v[i]), & s\geqslant w[i]\ \&\&\ t<w[i];\\ \max(m(i+1,s,t),m(i+1,s,t-w[i])+v[i],\\ \quad m(i+1,s-w[i],t)+v[i]), & s\geqslant w[i]\ \&\&\ t\geqslant w[i];\end{cases}$$

End.

4) 将算法转换成可执行程序.

有了上述递推关系,很容易得到该问题的算法程序,限于篇幅,程序省略.

## 3 背包问题的统一求解策略

对以上 3 个变形背包问题推导过程进行仔细分析,并将其推广应用到其它背包问题以及子集和问题、船装载问题的求解过程,得到了以下求解一类问题的统一策略.

1) 使用 0-1 数组形式化地描述问题的程序规约〈AQ,AR〉.

2) 根据问题的求解特征寻找结构和原问题相同但规模较小的子问题 $S$.

3) 使用范围分裂等量词的性质,对问题的后置断言 AR 进行等价变换,变换成如下形式:

$$AR=F(S,\Delta S)$$

其中 $S$ 是(1)式中获得的子问题,$\Delta S$ 是在子问题基础上求解原问题的增量部分.

4) 利用单一范围、一般分配律等量词性质,针对增量部分 $\Delta S$,分情形进行等价变换. 假设满足条件 $C_1$ 下,$\Delta S$ 等价变换为 $\Delta S_1$,满足条件 $C_2$ 下,$\Delta S$ 等价变换为 $\Delta S_2$,…,满足条件 $C_n$ 下,$\Delta S$ 等价变换为 $\Delta S_n$.

分情形变换时要求满足 $C_1\vee C_2\vee\cdots\vee C_n=\text{TRUE}$.

5) 结合 3) 和 4) 等价变换的结果,得到如下递推关系

$$AR=\begin{cases}F(S,\Delta S_1), & \text{当 } C_1 \text{ 成立时},\\ F(S,\Delta S_2), & \text{当 } C_2 \text{ 成立时},\\ \cdots & \\ F(S,\Delta S_n), & \text{当 } C_n \text{ 成立时}.\end{cases}$$

6) 确定递推的初始条件(即最小子问题的取值)及递推的终止条件(满足终止条件时即获得了原问题的解).

7) 首先定义变量记录递推关系中子问题的解,然后编写算法程序,根据(3)式中的结果从初始状态出发,利用循环结构和分支结构,基于(2)式中确定的递推关系,不断从规模较小子问题的解递推出规模较大子问题的解,直到满足终止条件,获得原问题的解.

## 4 总结

本文使用递推技术形式化推导了 3 个变形 0-1 背包问题,3 个问题的推导均使用了 0-1 数组表示问题的程序规约,在此基础上采用程序规约的变换技术,结合问题求解的不同情况对程序规约进行等

价变换,得到问题求解的递推关系,进而得到解决问题的算法程序. 本文中采用的程序规约表示方法以及求解递推关系的思路可以在其他背包问题以及子集和、船装载等问题的求解中加以推广应用. 该方法与传统算法设计方法相比:

1) 在对原问题分划的基础上,通过量词性质对程序规约进行等价变换得到问题求解的递推关系,进而得到问题算法,详细展示了算法推导过程.

2) 由于推导过程采用的是等价数学变换,因此本方法通过保证推导过程的正确性有效保证了最终算法的正确性; 更重要的是本文通过 3 个变形背包问题的求解得到解决一类问题的统一策略,使得形式化推导方法在面对复杂问题求解时,具有生成问题求解策略的能力.

今后将继续对更加复杂的背包问题以及传统算法问题( 如动态规划法、贪心法求解的问题) 进行深入研究,探索使用递推技术形式化推导复杂问题的有效途径.

## 5　参考文献


[1] 田烽楠,王于. 求解 0-1 背包问题算法综述 [J]. 软件导刊,2009,8(1):59-61.

[2] 王蔚,邱伟星. 整数的带余除法在子集和问题中的应用 [J]. 计算机工程,2011(s1):183-185.

[3] 刘玉娟,王相海. 0-1 背包问题的两种扩展形式及其解法 [J]. 计算机应用研究,2006,23(1):28-30.

[4] 程跃. 多背包问题的一种求解方法 [J]. 产业与科技论坛,2011,10(20):184-185.

[5] 孙瑞芳,焦晓君,施瑞娜,等. 分支限界装载问题的算法分析与设计 [J]. 软件设计开发,2015(3):72-76.

[6] 杨庆红,肖燕娟. 一种高效的算法程序设计方法: PAR 方法 [J]. 计算机与现代化,2000(6):1-5.

[7] 石海鹤,薛锦云. 一种基于 PAR 的高可靠算法程序设计技术 [C]. 第六届中国测试学术会议论文集,2010:433-437.

[8] 石海鹤,薛锦云. 基于 PAR 的算法形式化开发 [J]. 计算机学报,2009,32(5):982-991.

[9] Yang Qinghong. An algorithmic formalization mnthod based on recurrence technique [C]. The 8th International Conference on Computer Science & Education,2013.

[10] Xue Jinyun. A unified approach for developing efficient algorithmic programs [J]. Journal of Computer Science and Technology,1997,12(4):314-329.

[11] 胡启敏,薛锦云. 若干算法程序的形式化推导与生成技术研究 [J]. 计算机研究与发展,2008,45:148-153.

[12] 王昌晶,薛锦云. 算法及其时间复杂度可同步形式化推导的方法 [J]. 计算机应用研究,2008(3):681-683.

[13] 石海鹤,揭安全,薛锦云. 0-1 背包问题的一种新解法 [J]. 计算机工程,2008,34(17):37-49.

[14] 王昌晶,薛锦云. 一类 0-1 背包问题算法程序的形式化推导 [J]. 武汉大学学报: 理学版,2009,55(6):674-680.

[15] 石海鹤. 基于 PAR 的排序算法自动生成研究 [J]. 软件学报,2012(9):2248-2260.


## The Formal Derivation of Three Forms of 0-1 Knapsack Problems


YOU Ying, YANG Qinghong*, QI Leilei

(College of Computer Information Engineering, Jiangxi Normal University, Nanchang Jiangxi 330022, China)



**Abstract:** Based on depth study of various forms of knapsack problem in this paper, a vector of binary variables to describe program specification of various forms of knapsack problem formally has been used. Through program specification conversion technology, the recurrence relation of problem solving sequence which can solve various forms of knapsack problems has been used, based on which obtained their algorithm programs. Through this way, the reliability of the algorithm program is. guaranteed. The derivation methods that is used in this paper can also be applied in subset-sum problems, ship-loading problems and so on.



**Key words:** formal derivation; 0-1 knapsack problem; vector of binary; program specification; recurrence relation



( 责任编辑: 冉小晓)