# 算法设计与分析

**回溯法解决多维背包问题**

**学号：201707406154 班别：计算机172班 姓名：刘倩羽**

**问题描述：**给定n种物品和一背包，每个物品有m个属性，给定背包对每个属性的约束条件,求装入背包中物品的最大总价值。

**问题分析：**解多维背包问题和解0-1背包问题非常相似，多维背包问题是常规0-1背包问题的扩展问题，解决多维背包问题仍要基于0-1背包问题的基础上进行求解，物品仍有放与不放两种选择，最终目标仍是求解出使得物品总价值最大的选择。用回溯法需要构造解的子集树。

**算法设计：**

1. 设n为物品个数，m为物品约束条件个数；bestp为当前最优总价值（最优解）;cp为当前总价值；分别用p[i]存储第i种物品的价值,a[i] [j] 存放物品属性约束条件,b[i] 存放约束条件的总值，bestpcp[k] 存放目前最大价值；

b. 用递归函数Backtrack (i,b)来实现回溯法搜索子集树（形式参数i表示递归深度，n用来控制递归深度，形式参数cp当前总价值，bestp表示当前最优总价值）：

1.若i >=n，则算法搜索到一个叶结点，判断当前总价值是否最优： 若cp>bestp，更新当前最优总价值为当前总价值（即bestp=cp），更新装载方案（即bestcp[k]=bestp）；

2.采用for循环对物品i放与不放两种情况进行讨论（0≤j≤1）：

若i<n,，如果s=0，则中止，跳出当前循环；如果s==1,获取当前更新价值(cp=cp+p[i])有m个约束条件循环m次for（K=1;K<m;K++）;存放物品后减去改物品的容量，即b[K]=b[K]-a[K][i]，然后调用一次进行回溯Backtrack(i+1,b)然后获取当前价值。即cp=cp-p[i]。

3.若有m个约束条件则循环m次， for(h=1;h<=m;h++）；函数Backtrack(i+1,b)调用结束后则返回当前总价值和背包容量。

1. 主函数调用一次backtrack(1,b)即可完成整个回溯搜索过程，最终得到的bestp和bestx[i]即为所求最大总价值和最优装载方案。

源码

#include<stdio.h>

int n;//n是物品的个数

int m;//m是物品的约束条件个数

int s,j,k,h,z;//用于循环判断

float p[10000];//定义物品的价值

float a[10000][10000]; //物品属性条件

float b[10000];//物品约束总值

float bestp=0,cp=0;//bestp表示当前最优总价值,cp表示当前总价价值；

float bestcp[10000];//目前最大价值，将bestp的值存入bestpcp[]；

void Backtrack(int i,float b[])

{

if(i>n)//回溯结束,算法搜索到叶子结点

{

if(cp>bestp)//当前总价值大于目前最优总价值

bestp=cp;//将CP赋值给bestp，即更新当前最优总价值为当前总价值

k++;

bestcp[k]=bestp;//将bestp暂时存在一个数组

}

else

{ s=1;

for(j=1;j<=m;j++)

{

z=b[j]-a[j][i]; //判断物品是否满足条件

if(z<0){

s=0; //s=0表示不选此物品

break;}}

if(s==1) //s==1，表示选此物品

{cp=cp+p[i]; //存放物品后，获得当前价值cp

for(k=1;k<=m;k++) //有m个约束条件就循环m次

{

b[k]=b[k]-a[k][i]; //存放后减去容量

}

Backtrack(i+1,b);

cp=cp-p[i]; //到达叶子结点后，回溯

for(h=1;h<=m;h++){ //有m个约束条件就循环m次

b[h]=b[h]+a[h][i]; //拿出已经存放的物品后，加上它原来的容量

}

}

Backtrack(i+1,b);

}

}

void main()

{

int i,j;

printf("请输入物品个数和约束条件个数:\n");

scanf("%d %d",&n,&m);

printf("请依次输入物品的价值:\n");

for(i=1;i<=n;i++)

scanf("%f",&p[i]);

printf("请输入物品属性条件:\n");

for(i=1;i<=m;i++)

for(j=1;j<=n;j++)

scanf("%f",&a[i][j]);

printf("请输入约束条件的总值\n");

for(i=1;i<=m;i++)

scanf("%f",&b[i]);

Backtrack(1,b);

printf("最大价值为:\n");

printf("%.1f\n",bestcp[k-1]);

}