暨南大学本科实验报告专用纸

课程名称 算法分析与设计实验 成绩评定

实验项目名称 最小重量机器设计 指导教师 李展

实验项目编号 实验 X 实验项目类型 综合性 实验地点

学生姓名 张印祺 学号 2018051948

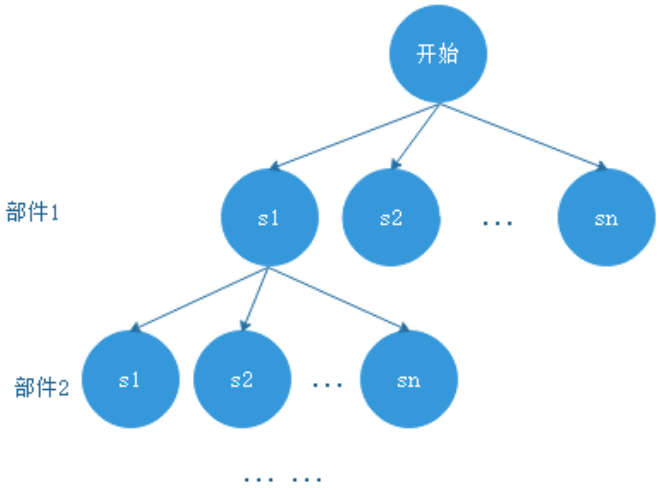
学院 信息科学技术 系 计算机科学 专业 网络工程

实验时间 2020 年 5 月 20 日

一、问题描述

设某一机器由 n 个部件组成，每一种部件都可以从 m 个不同的供应商处购得。设w_{ij}是供应商 j 处购得的部件 i 的重量，cij 是相应的价格。试设计一种算法，给出总价格不超过 c 的最小重量机器设计。

二、算法思路

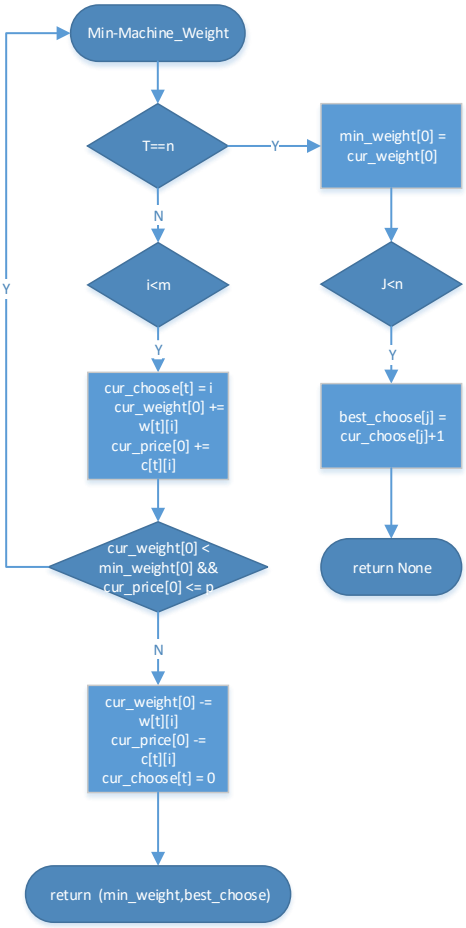


在算法开始时将部件一对应的所有供应商加入队列，然后对队列中进行以下操作：

利用回溯法求解问题，首先可以为该题的二维数组 w、c、n、m、p 赋初值，构造回溯法中的解空间树

根据回溯法的要求，进行的是深度遍历，同时满足重量尽可能小，价格不能超过 p。

三、流程图



四、测试结果

```
# 限定：假设有解
n = 3 # 零件数量
m = 3 # 供应商数量
p = 4 # 不应超过的价格
w = [[1,2,3], [3,2,1], [2,2,2]] # 供应商j处购得的部件 i 的重量
c = [[1,2,3], [3,2,1], [2,2,2]] # 相应的价格

Min_Machine_Weight(n,m,p,w,c)
```

```
最小的重量是:4
最佳选择是: [1, 3, 1]

<__main__.Min_Machine_Weight at 0x2595cf92ef0>
```

```
# 限定：假设有解
n = 4 # 零件数量
m = 3 # 供应商数量
p = 10 # 不应超过的价格
w = [[2,1,4], [3,2,3], [4,3,2],[5,4,1]] # 供应商j处购得的部件 i 的重量
c = [[5,4,1],[4,3,2],[3,2,3],[2,1,4]] # 相应的价格

Min_Machine_Weight(n,m,p,w,c)
```

```
最小的重量是:10
最佳选择是: [2, 2, 2, 2]

<__main__.Min_Machine_Weight at 0x2595d00c5f8>
```

五、实验总结

利用分支限界法，每次选择子结点都要进行分支判断。

假设不考虑价格限制，每个部件在哪购买都是相互独立的，对于每个部件只需要考虑提供最轻部件的供应商。考虑上价格限制后，除了考虑质量最轻，还要考虑购买这个部件后是否还有足够的经费购买其他部件。因此要先记录总的最低经费，购买完一个部件后便比较一次剩余经费是否低于剩余最低经费需求，如果是，则经费不足以购买全部部件，该结点不进行展开。

T(n) = o(m^n)。

六、源代码

```
class Min_Machine_Weight():
    def backtrack(self,t,n,m,p,w,c,cur_weight, cur_price, cur_choose, mi
n_weight,best_choose):
    # 此处的 t 代表每一次遍历 i 供应商的零件
    #    global cur_weight, cur_price, cur_choose, min_weight, p, w, c
        if t == n:
            # 遍历到叶子结点
            if cur_weight[0] < min_weight[0]:
                min_weight[0] = cur_weight[0]

                for j in range(n):
                    if cur_choose[j]==None:
                        cur_choose[j]=0
                    best_choose[j] = cur_choose[j]+1
                return

        else:
            for i in range(m): # 遍历供应商
                if cur_choose[t]==None:
                    cur_choose[t]=0
                cur_choose[t] = i
                cur_weight[0] += w[t][i]
                cur_price[0] += c[t][i]

                if cur_weight[0] < min_weight[0] and cur_price[0] <= p:
                    # 该供应商的重量小于局部最优解 同时价格满足要求 则遍历其子树
                    self.backtrack(t+1,n,m,p,w,c,cur_weight, cur_price,
cur_choose, min_weight,best_choose)

                cur_weight[0] -= w[t][i]
                cur_price[0] -= c[t][i]
```

```
                cur_choose[t] = 0
        return  (min_weight,best_choose)
    def __init__(self,n,m,p,w,c):
        best_choose = [None for i in range(n)]
        cur_choose = [None]*1000
        cur_weight = [0]
        cur_price = [0]
        min_weight = [10000]
        min_weight,best_choose=self.backtrack(0,n,m,p,w,c,cur_weight, cu
r_price, cur_choose, min_weight,best_choose)
        print('最小的重量是:%d'%min_weight[0])
        print('最佳选择是:',best_choose)
```