Knapsack Problem

Branch & Bound Algorithm

University of Vienna

Class: Computational Optimisation
Prof. Stefan Rath

Sylwia Genowefa Pytko
11816730

16.01.2018

Structures

Item:

- id
- profit
- weight
- profitPerWeight

Node:

- id
- level
- includeArray
- lowerBound
- cost (negated profit) - relaxed problem

Solution form

IncludeArray:

- 1 - Item included
- 0 - Item not included
- -1 - Item not concidered

Pseudocode - Initial steps

1. sort(Items) by profit/weight
2. for each profit in Item: profit = 0 - profit
3. create(RootNode)
    a. level = -1
    b. initiate IncludeArray with -1
    c. cost = 0
    d. lowerBound = 0
4. upperBound =0
5. nodeVector.push(RootNode)

Pseudocode - Iterating through Nodes

```
while(nodeVector.notEmpty()){
        sort(nodeVector) by cost
        currentNode<- bestNode
        If(currentNode is not last level){
                Branch(currentNode)
        }
}
```

Pseudocode - Branching

```
newNode.level = currentNode.level+1

includeArray[newNode.level] = 1 or includeArray[newNode.level] = 0

if (calculate(guaranteedWeigh) <= capacity){

        findNextItems()

}

else{

        kill(newNode)

}
```

Pseudocode - Find Next Items

```
while(newWeight<capacity){
        weight += nextItem.weight
        cost +=nextItem.profit
}
lowerbound  = cost
cost += fraction(nextItem.profit)
```

Pseudocode - Update Upperbound

```
if(upperbound>= newNode.lowerBound){
        upperbound = newNode.loweBound
        bestIncludeArray = includeArray
        for every node in nodeVector{
                if(node.cost> upperBound){
                        kill(node)
}
```

Printing solution

```
If (mySolution== fileSolution){
        print(solution”)
}
else{
        print(“failed”)
```

Output

p01 size: 10 time: 0.000s solution: 309 items: 1111010000

p02 size: 5 time: 0.000s solution: 51 items: 01110

p03 size: 6 time: 0.000s solution: 150 items: 110010

p04 size: 7 time: 0.000s solution: 107 items: 1001000

p05 size: 8 time: 0.000s solution: 900 items: 10111011

p06 size: 7 time: 0.001s solution: 1735 items: 0101001

p07 size: 15 time: 0.010s solution: 1458 items: 101010111000011

Computational complexity

Number of nodes in full binary tree of depth k

$$2^{k+1} - 1$$

File	Size	Num of Nodes (Full Tree)	Time
p06	7	256	0.001s
p07	15	65 536	0.018s
p08	24	33 554 432	unknown