File - V:\Teaching\AOM-BDAuU\Spring2017\Code\Bi-knapsack\bi-knapsack.py

```
# Python code to solve problem with two knapsack-like constraints
#   by dynamic programming

#  Helper functions for readability
#  Weight is the first element of each state, volume is the second
def weight(i) :
    return i[0]
def volume(i) :
    return i[1]

hugeNumber = float("inf")

items = 4
weightCapacity = 300
volumeCapacity = 200
itemBenefit  = ['unused',800,150,300,500]
itemWeight   = ['unused', 40,  8, 20, 15]
itemVolume   = ['unused', 10, 12, 22,  5]
numAvailable = ['unused',  3, 10,  4,  5]
stages = items

f = [dict( ) for t in range(stages + 2)]
x = [dict( ) for t in range(stages + 1)]

initialState = (weightCapacity,volumeCapacity)

# New component: reachable[t] is the set of states reachable in stage t

reachable = [set() for t in range(stages+2)]
reachable[1].add(initialState)
for t in range(1,stages+1) :
    for i in reachable[t] :
        maxCanInsert = min(numAvailable[t],
                           int(weight(i)/itemWeight[t]),
                           int(volume(i)/itemVolume[t]))
        for d in range(maxCanInsert+1) :
            reachable[t+1].add((weight(i)-d*itemWeight[t],volume(i)-d*itemVolume[t]))
    print(str(len(reachable[t+1])) + " different states reachable at stage " +
          str(t+1))

for i in reachable[stages + 1] :
    f[stages+1][i] = 0

for t in range(stages,0,-1) :

    for i in reachable[t] :

        maxCanInsert = min(numAvailable[t],
                           int(weight(i)/itemWeight[t]),
                           int(volume(i)/itemVolume[t]))
        value = -hugeNumber

        for d in range(maxCanInsert+1) :
            j = (weight(i)-d*itemWeight[t],volume(i)-d*itemVolume[t])
            moveValue = d*itemBenefit[t] + f[t+1][j]
            if moveValue > value :
                value = moveValue
                bestMove = d

```

```
60          # End of d loop
61
62          f[t][i] = value
63          x[t][i] = bestMove
64
65      # End of i loop
66
67  # End of t loop
68
69  print("Optimal value is " + str(f[1][initialState]))
70
71  solutionString = "Item counts :"
72  i = initialState
73  for t in range(1,stages+1) :
74      d = x[t][i]
75      solutionString += " " + str(d)
76      i = (weight(i)-d*itemWeight[t],volume(i)-d*itemVolume[t])
77
78  print(solutionString)
79  print("Total weight = " + str(weightCapacity - weight(i)) +
80        ", total volume = " + str(volumeCapacity - volume(i)))
81
```