CS-538 Combinatorial Optimization Project Part-3

Team Members

Archi Dsouza, Abhishek Bhardwaj, Piyush Nath

Given Instance : An element set E. A collection of subsets of E, S ⊆ 2 E with ∪B∈SB = E. Here 2E, sometimes denoted as P(E), is the set of all subsets of E. A cost cB for each subset B ∈ S. A covering requirement re for each element e ∈ E. A non-negative integer P.

Required Solution A sub-collection S′ ⊆ S such that at least P elements in E are fully covered by S′ i.e.|{e ∈ E : e ∼ S′}| ≥ P. Definition of covering: Let S′ e = {B ∈ S′: e ∈ B} be the collection of those sets in S′ containing e.An element e ∈ E is said to be fully covered by S′ if |S′ e| ≥ re, denoted by e ∼ S′.

Measure A sub-collection S ′ with the minimum cost. We define the cost of a sub-collection S ′ to be c(S ′) = ΣB∈S′c(B). We have devised the following Integer Linear Program for PSMC (called IP): Minimize X B∈S cBxB subject to X e∈E ye ≥ P (1) X B∈S : e∈B xB – reye ≥ 0 ∀ e ∈ E (2) xe ≥ 0 ∀ B ∈ S (3) xe ≤ 1 ∀ B ∈ S (4) ye ≤ 1 ∀ e ∈ E (5) ye ≥ 0 ∀ e ∈ E (6) xe ∈ Z ∀ B ∈ S (7) ye ∈ Z ∀ e ∈ E (8) The variables xB are meant to be 1 iff B ∈ S′ and ye are meant to be 1 iff e is chosen to be multi-covered.

In [192]:

```
#Reading The Input File
with open("intance01.txt","r") as f:
    content = f.readlines()

#Striping The Lines of The Input File
content = [x.strip().split(" ") for x in content]

#E=Cardnality of Set E,S=Cardnality of Subset, P = Positive Inteager
[E,S,P] = [int(x) for x in content[0]]
subsetlist=[]
#Loading Subsets From file
for subset in content[3:]:
    subsetlist.append(tuple(int(x) for x in subset))

#Loading RE From file which is covering requirement
RE_values = [ int(x) for x in content[1] ]

#Loading Cost From file
Cost_values = [ int(x) for x in content[2] ]
```

In [185]:

```
#Intilaizing GLPK and Pulp for LP Solving
import pulp
pulp.pulpTestAll()
problem = pulp.LpProblem("MinimumProblem", pulp.LpMinimize)
```

```
        Testing zero subtraction
        Testing inconsistant lp solution
        Testing continuous LP solution
        Testing maximize continuous LP solution
        Testing unbounded continuous LP solution
        Testing Long Names
        Testing repeated Names
        Testing zero constraint
        Testing zero objective
        Testing LpVariable (not LpAffineExpression) objective
        Testing Long lines in LP
        Testing LpAffineExpression divide
        Testing MIP solution
        Testing MIP solution with floats in objective
        Testing MIP relaxation
        Testing feasibility problem (no objective)
        Testing an infeasible problem
        Testing an integer infeasible problem
        Testing column based modelling
        Testing dual variables and slacks reporting
        Testing fractional constraints
        Testing elastic constraints (no change)
        Testing elastic constraints (freebound)
        Testing elastic constraints (penalty unchanged)
        Testing elastic constraints (penalty unbounded)
* Solver <class 'pulp.solvers.PULP_CBC_CMD'> passed.
Solver <class 'pulp.solvers.CPLEX_DLL'> unavailable
Solver <class 'pulp.solvers.CPLEX_CMD'> unavailable
Solver <class 'pulp.solvers.CPLEX_PY'> unavailable
Solver <class 'pulp.solvers.COIN_CMD'> unavailable
Solver <class 'pulp.solvers.COINMP_DLL'> unavailable
        Testing zero subtraction
        Testing inconsistant lp solution
        Testing continuous LP solution
        Testing maximize continuous LP solution
        Testing unbounded continuous LP solution
        Testing Long Names
        Testing repeated Names
        Testing zero constraint
        Testing zero objective
        Testing LpVariable (not LpAffineExpression) objective
        Testing LpAffineExpression divide
        Testing MIP solution
        Testing MIP solution with floats in objective
        Testing MIP relaxation
        Testing feasibility problem (no objective)
        Testing an infeasible problem
        Testing an integer infeasible problem
        Testing column based modelling
        Testing fractional constraints
        Testing elastic constraints (no change)
        Testing elastic constraints (freebound)
        Testing elastic constraints (penalty unchanged)
        Testing elastic constraints (penalty unbounded)
* Solver <class 'pulp.solvers.GLPK_CMD'> passed.
Solver <class 'pulp.solvers.XPRESS'> unavailable
```

```
Solver <class 'pulp.solvers.GUROBI'> unavailable
Solver <class 'pulp.solvers.GUROBI_CMD'> unavailable
Solver <class 'pulp.solvers.PYGLPK'> unavailable
Solver <class 'pulp.solvers.YAPOSIB'> unavailable
```

In [186]:

```
#Setting the lower bound and upper bound for variables in GLPK Class
X = pulp.LpVariable.dicts('X_B', range(S), lowBound = 0,upBound = 1,cat = pulp.Lp
Y = pulp.LpVariable.dicts('Y_E', range(E), lowBound = 0,upBound = 1,cat = pulp.Lp
```

In [187]:

```
#Assigning the Objective function to the GLPK Class
problem+= pulp.LpAffineExpression(list(zip(X.values(),Cost_values))), "The object
```

In [188]:

```
for i in range(1,E+1):
    temp = []
    for index,subset in enumerate(subsetlist):
        if i in subset:
            temp.append((X.get(index),1))
    problem+= pulp.LpAffineExpression(temp)-RE_values[i-1]*Y.get(i-1) >=0,"{0} co
```

In [189]:

```
problem+= pulp.LpAffineExpression([(Y.get(i),1) for i in range(E)]) >= P,'P value
if problem.solve()== 1:
    print("Problem Solved")
    print("Minimum Cost : ",pulp.value(problem.objective))
```

```
Problem Solved
Minimum Cost :  48.0
```

In [190]:

```
#Identifying the subsets used to get the optimal cost.
counter = 0
index_list = []
for variable in problem.variables()[:E]:
    counter += 1
    if variable.varValue==1:
        index_list.append(counter)
```

In [191]:

```
#Creating The Solution File
print("Result:",str(len(index_list))+" "+str(int(pulp.value(problem.objective)))+
result = str(len(index_list))+" "+str(int(pulp.value(problem.objective)))+" "+' '
with open ('solution01.txt','w') as f:
    f.write(result)
print("solution01.txt created")
```

```
Result: 3 48 1 3 5
solution01.txt created
```

In [ ]:

```
```