IE - 501 : Optimization Models

Homework Assignment - 2

Instructed by: Avinash Bhardwaj

Name: Shivam Tiwari

Roll Number: 23N0463

Department : IEOR

Contents

NOTE:
I have discussed with my teammates namely Dheeraj(22b0935), Shivam Tiwari(23n0463) and Shivam Gupta(2n0460) while solving the problems, and I assure that it was not something as copying but solely a group discussion about various approaches.

1 Question - 1

Our project title is "**Optimizing Cloud Resource Allocation for Performance and Cost Efficiency**"

1. **Motivation for the project problem**
   - The main motivation behind this idea is increasing applications of cloud computing (Basically increasing demand :)
   - Organization and customers need cost-effective and better performance.
2. **Detailed description of the problem**
   - Cloud computing is a area where efficient usage of resources plays a key role
   - Key objectives include minimizing cost, maximizing the performance.
   - We also want to predict the cost for different possible resource and customer combinations
3. **Tentative solution methodology ideated**
   - We focused on parameters and constraints for choosing this project.
   - Key parameters are the computing resources needed for a workload, including CPU, memory, storage, and network bandwidth.
   - Other parameters are Budget and minimum performance metrics.
   - Besides the trivial constraints such as resource, budget, time etc.. other possible constraints are security and network constraints.
   - However the main objective is to maximize the performance and minimize the cost
4. **Tentative deliverables**
   - We want to finally design a strategy and give a algorithm to solve the problem
   - We will also try to give outcomes for some possible combinations of inputs
5. **Distribution of work amongst team members**
   - For now we haven't distributed the work but we hope we will be doing that as soon as possible and start the project work.

2 Question - 2

The actual Farkas Lemma is given by

Let A $\in \mathbf{R}^{mn}$ and b $\in \mathbf{R}^m$ then exactly of the following two are correct

1. $\exists$x $\in \mathbf{R}^n$, Ax = b and x *geq* 0.
2. $\exists$p $\in \mathbf{R}^m$, p^{T}A $\geq$ 0 and p^{T}b $<$ 0.

The above is certificate of infeasibility for standard form of optimization problem.

Now we will define certificate of infeasibility for canonical form i.e.,
Let A $\in \mathbf{R}^{mn}$ and b $\in \mathbf{R}^m$ then exactly of the following two are correct

1. $\exists$x $\in \mathbf{R}^n$, Ax $\leq$ b.
2. $\exists$p $\in \mathbf{R}^m$, p $\geq$ 0 and p^{T}A = 0 and p^{T}b $<$ 0.

Now we will prove the second one.

- **Step-1 :** Proving atmost one is true
  Let both 1) and 2) be true.
  2) is true, $\implies$ $\exists$ p $\geq$ 0 s.t p^{T}A = 0 - (I) and p^{T}b $<$ 0.
  1) is true, $\implies$ $\exists$ x s.t Ax $\leq$ b - (II)

  multiplying p^T to (II) on both sides,

  $$p^T Ax \leq p^T b$$

  from (I) we get

  $$0 \leq p^T b$$

  Contradiction. Therefore both of them can't be true simultaneously.

- **Step-2 :** Proving atleast one is true
  Let both 1) and 2) be false. 1) false $\implies$ there is no solution for Ax $\leq$ b. We don't know about sign of x, so we manipulate such that

  $$\implies \text{There is no solution for} \quad A(x^+ - x^-) \leq b \quad \text{where } x^+, x^- > 0$$

  $$\implies \text{There is no solution for} \quad A(x^+ - x^-) + Is = b \quad \text{where } x^+, x^-, s > 0, s \in \mathbf{R}$$

  So we can write this as,

  $$A' = [A \quad -A \quad I] \quad x' = [x^+ \quad x^- \quad s]^T$$

  $$\implies \text{There is no solution for} \quad A'x' = b$$

  From Farkas lemma, we can infer that as 1) is false 2) will be true

  $$\implies \exists p \text{ such that } p^T A' \geq 0 \text{ and } p^T b < 0$$

  $$\implies \exists p \text{ such that } p^T A \geq 0 \text{ and } p^T A \leq 0 \text{ and } p \geq 0 \text{ and } p^T b < 0$$
  $$\implies \exists p \text{ such that } p \geq 0 \text{ and } p^T A = 0 \text{ and } p^T b < 0$$

  Contradiction. So we can infer that atleast one of them is true.

In this way we can derive the certificate of infeasibility for given (LP).

3 Question - 3

For this question, I have used the idea of Hamiltonian circuit of a graph.
For this first I have seen the map given and selected 40 nodes. Then I have manually drawn a graph as shown below and measured distances using google maps.

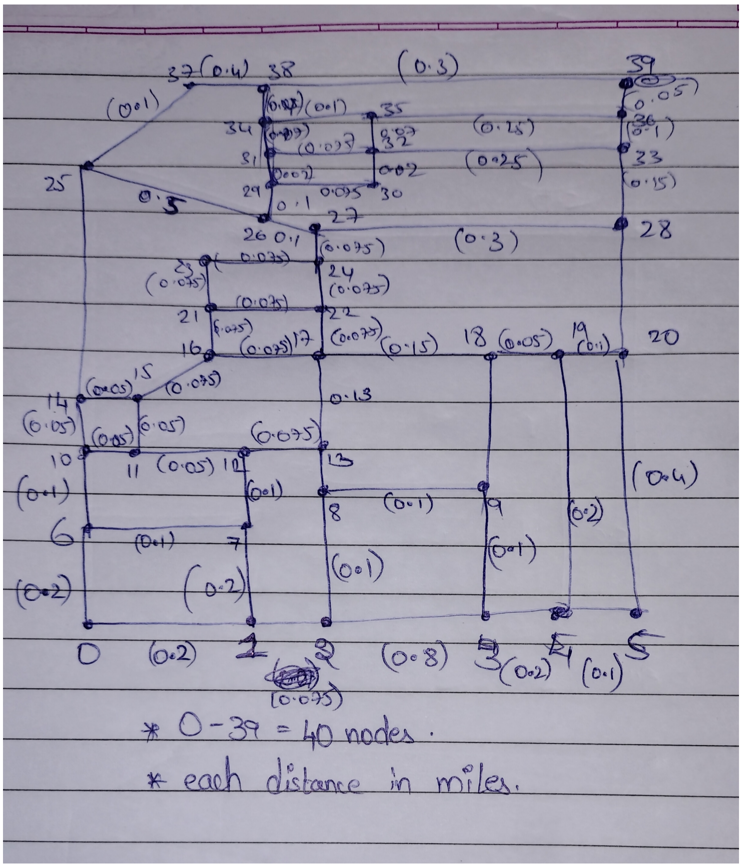

Figure 1: Handwritten graph

So Now I have a graph. To start with formulation,
Parameters

- The set of vertices **V** and set of edges **E** i.e., the Graph G(V,E)
- We also have length of each edge. Let us call it d_{ij} for all (i, j) ∈ E.

Decision variables

-
$$x_{ij} = \begin{cases} 1, & \text{if (i, j)} \in \mathbf{E} \\ 0, & \text{otherwise} \end{cases}$$

- u_i is a real number for each i ∈ V.

Objective function

- We want to maximize total distance covered i.e.,

$$\sum_{i=0}^{39}\sum_{j=0}^{39} d_{ij}x_{ij} \quad \text{has to be maximized}$$

Constraints

- We know that in Hamiltonian circuit each node will be involved in exactly 0 or 2 edges
- The first constraint is that any node has to be visited exactly once i.e.,

$$\sum_{i=0}^{39} x_{ij} = 1$$

- The first constraint is that any node has to be left exactly once i.e.,

$$\sum_{j=0}^{39} x_{ij} = 1$$

- The third constraint is the difficult one. To maintain no subtours. This is done using below equation

$$u_i - u_j + x_{ij} \leq (n-1) * (1 - x_{ij})$$

  This is actually called Miller-Tucker-Zemlin subtour elimination constraints. This is also used in travelling sales man problem to eliminate subtours.

  Let us see how does MTZ work.
  If $x_{ij} = 1$, then $u_i - u_j \leq -1$
  If $x_{ij} = 0$, then $u_i - u_j \leq n - 1$

  This u_i actually gives the position of vertex i in the hamiltonian circuit (increasing order). We intentionally put $u_0 = 0$, so it is the starting point. So This ranking of vertices make sure that any vertex is involved in the tour involving vertex 0 or it doesn't involve in the circuit. Therefore there are no subtours.

Programming part

- After formulating as above I have used "Pyomo" modelling tool and "ipopt" as solver.
- My code is as shown below in figure 2,
- The output of this code is "Optimized Objective Value : 8232.001534212483 meters" i.e., 8.23 kilometers

```python
from __future__ import division
from pyomo.environ import *

num_nodes = 40

edges = [(0, 1, 200), (0, 6, 200), (1, 2, 75), (1, 7, 200), (2, 3, 80), (2, 8, 100), (3, 4, 200), (3, 9, 100), (4, 5, 100), (4, 19, 200),
 (5, 20, 400), (6, 7, 100), (6, 10, 100), (7, 12, 100), (8, 9, 100), (8, 13, 200), (9, 18, 200), (10, 11, 50), (10, 14, 50), (11, 12, 50),
 (11, 15, 50), (12, 13, 75), (13, 17, 130), (14, 15, 50), (14, 25, 200), (15, 16, 75), (16, 17, 75), (16, 21, 75), (17, 18, 150), (17, 22, 75), (18, 19, 50),
 (19, 20, 100), (20, 28, 250), (21, 22, 75), (21, 23, 75), (22, 24, 75), (23, 24, 75), (24, 27, 75), (25, 26, 500), (25, 37, 100), (26, 27, 100), (26, 29, 100),
 (27, 28, 300), (28, 33, 150), (29, 30, 75), (29, 31, 20), (30, 32, 20), (31, 32, 75), (31, 34, 70), (32, 33, 250), (32, 35, 70), (33, 36, 100), (34, 35, 100),
 (34, 38, 70), (35, 36, 250), (36, 39, 50), (37, 38, 400), (38, 39, 300)]

model = ConcreteModel()

model.x = Var(range(num_nodes), range(num_nodes), within=Binary)
model.u = Var(range(num_nodes), within=NonNegativeReals)

def obj_rule(model):
    sum = 0
    for edge in edges:
        sum = sum + edge[2] * (model.x[int(edge[0]),int(edge[1])])
    return sum

def visit_rule(model,i):
    sum = 0
    for j in range(num_nodes):
        sum = sum + model.x[i,j]
    return sum == 1

def leave_rule(model,j):
    sum = 0
    for i in range(num_nodes):
        sum = sum + model.x[i,j]
    return sum == 1

# def mtz_constraints_rule(model, i):
#     if i != 0:
#         return sum(model.x[i, j] for j in range(num_nodes) if j != i) == 1 - model.u[i]
#     else:
#         return Constraint.Skip

def mtz_constraints_rule(model, i, j):
    if i != 0:
        return model.u[i] + model.x[i,j] <= model.u[j] + (num_nodes - 1) * (1 - model.x[i,j])
    else:
        return Constraint.Skip

model.obj = Objective(rule=obj_rule, sense=maximize)
model.con1 = Constraint(range(num_nodes), rule=visit_rule)
model.con2 = Constraint(range(num_nodes), rule=leave_rule)

model.con3 = Constraint(range(num_nodes), range(num_nodes), rule=mtz_constraints_rule)

opt = SolverFactory("ipopt", executable='/opt/homebrew/bin/ipopt', validate=False)
result = opt.solve(model)

if result.solver.status == SolverStatus.ok and result.solver.termination_condition == TerminationCondition.optimal:
    print("Optimized Objective Value : ", (model.obj())*1.6, " meters")
else:
    print("Optimization was not successful.")
```

Figure 2: Code