THE UNIVERSITY of ADELAIDE

Advance Algorithms

Assignment on Linear programming with inequalities

Manivannan Meenakshi sundaram

Uni Adeladie

a1842066

Due Date : May 13, 2022

Assignment 2

Exercise 1 :

Objective Function :

2 max{a + b + c, 1} ≤ h(a, b) + h(a, c) + h(b, c) ≤ 7 /3 max{a + b + c, 1}

Additional Functions Required :

h(a, b) = 5/ 6 max{a + b, 1} + 2 /3 max{a + b, 1/2 } − 1 /3 max{a, b, 1/2 } – 1/ 3

h(a, c) = 5 /6 max{a + c, 1} + 2 /3 max{a + c, 1/2 } − 1 /3 max{a, c, 1/2 } – 1/ 3

h(b,c) = 5 /6 max{b + c, 1} + 2 /3 max{b + c, 1/2 } − 1 /3 max{b, c 1/2 } – 1/ 3

Explanation :

To construct a linear programming model with math inequalities, we need to construct a model with variables such as a,b,c and also we need to induce them with auxiliary variables called x,y,z and few more while we onboarding new expressions.

We introduce,

P = max{a+b+c,1}

Hx1 = max {a+b,1}	Hx2 = max {a+c,1}	Hx3 = max{ b+c,1}
Hy1 = max{a+b,1/2}	Hy2 = max{a+c,1/2}	Hy3 = max{a+c,1/2}
Hz1= max{a,b,1/2}	Hz1= max{a,c,1/2}	Hz3= max{b,c,1/2}

We now find the equation to the following with the considerations from above as,

2p ≤ Hab + Hac +Hbc ≤ 7/3 P

In this assignment, we will solve for both equations where the terms are maximized and minimized to 0 with respect to lower and upper bound variables.

Finding constrains for 2 agents :

P= max {a+b+c, 1}

P >= a+b+c ----1

P >= 1 ----2

P <= a+b+c + 10000* Pprime ----3

P <= 1 + 10000*(1-Pprime) ----4

These are constrains that are built for 2 agents, in this Pprime acts as a switch which switches between 0, 1 as they are binary variables.

If Pprime == 0, sub in equation 3 and 4

We get,

p<=a+b+c

p<=1+ 10000 => This is considered as a larger variable, hence neglect it.

So, equating the equations

p<=a+b+c and p >=a+b+c , we get

p= a+b+c

if Pprime==1, sub in equation 3 and 4

we get,

p<=1

p<=a+b+c+10000 => This is considered as a larger variable, hence neglect it.

So, equating the equations

P<=1 and p>=1 , we get

P=1

Similarly this can be solved for the other equations which is having the constrain for 2 agents.

Finding constrains for 3 agents :

Hx3= max{a,b,1/2}

Hx3 >= a ----1

Hx3 >= b ----2

Hx3 >= ½ ----3

Hx3 <= a + 10000* Hx3prime1 ----4

Hx3<= b+10000*Hx3prime2 ----5

Hx3 <= 1/2 + 10000*(1- Hx3prime3) ----6

These are the constrains built for 3 agents, in this prime acts as a switch which juggles between 0, 1 as it is a binary variables.

We consider binary variables for three agents as follows

- 1 0 0
- 0 1 0
- 0 0 1

With the above representations of binary values the constrains are designed in such a way that only one of the three agents will have the max value.

If Hx3prime ==1 for eq 4, We induce Hx3prime ==0 for others in equation 5,6

We get,

Hx3<=a+10000=> This is considered as a larger variable, hence neglect it.

Hx3x<=b

Hx3<=1+10000=> This is considered as a larger variable, hence neglect it.

So equating this, we get

Hx3 = b

If Hx3prime ==1 for eq 5, We induce Hx3prime ==0 for others in equation 4,6

We get,

Hx3<=a

Hx3x<=b+10000 => This is considered as a larger variable, hence neglect it.

Hx3<=1+10000=> This is considered as a larger variable, hence neglect it.

So equating this, we get

Hx3 = a

If Hx3prime ==1 for eq 6, We induce Hx3prime ==0 for others in equation 4,5

We get,

Hx3<=a, these has been already equated in the previous substitutes

Hx3x<=b , these has been already equated in the previous substitutes

Hx3<=1/2

So equating this, we get

Hx3 = ½

Objective Function to Maximize 0 :

2p ≤ Hab + Hac +Hbc ≤ 7/3 P

y= Hab + Hac +Hbc

2p <= y <= 7/3p

2p – y and y- 7/3 p . If this is equal to zero, that's out proof for upper bound.

Value obtained are A= 0.5, B= 0, C= 0.5.

Substituting these in the equation we get equations as

2 <= 2.2 <= 2.3, which proves that our solutions are optimal and the values of A,B,C are with in the range of 0 and 1

```
Maximizeprob1 = 2*p -y
Maximizeprob2= y - (7/3 * p)

prob += (Maximizeprob1 and Maximizeprob2)
```
✓ 0.9s

```python
    prob.solve()
    print("Status:", LpStatus[prob.status])
    print(a.name, "=", a.value())
    print(b.name, "=", b.value())
    print(c.name, "=", c.value())
    print("Objective value for upper bound = ", value(prob.objective))
```

✓ 0.4s

```
Status: Optimal
A = 0.5
B = 0.0
C = 0.5
Objective value for upper bound =  0.0
```

Objective function to Minimize 0 :

2p ≤ Hab + Hac +Hbc ≤ 7/3 P

y= Hab + Hac +Hbc

7/3p-y and y-2p. If this is equal to zero, that's out proof for lower bound.

Value obtained are A= 0.5, B= 0.5, C= 0.5.

Substituting these in the equation we get equations as

3 <= 4 <= 7, which proves that our solutions are optimal and the values of A,B,C are with in the range of 0 and 1

```python
Minimizeprob1 = (7/3 * p) - y
Minimizeprob2 = y - 2 * p

# Maximizeprob1 = 2*p -y
# Maximizeprob2= y - (7/3 * p)

prob += (Minimizeprob1 and Minimizeprob2)
```

```python
    prob.solve()
    print("Status:", LpStatus[prob.status])
    print(a.name, "=", a.value())
    print(b.name, "=", b.value())
    print(c.name, "=", c.value())
    print("Objective value for lower bound = ", value(prob.objective))
```

✓ 0.3s

```
Status: Optimal
A = 0.5
B = 0.5
C = 0.5
Objective value for lower bound =  0.0
```

Appendix :

- Some of the constrains that I have added to solve this.
- These are some of the unique constrains which has to be added, rest of the constrains for the equation can be replicated accordingly based on the equations.

```python
prob += hx1 >= a + b
prob += hx1 >= 1
prob += hx1 <= a + b + 1000000*hxPrime1
prob += hx1 <= 1 + 1000000*(1-hxPrime1)
```

✓ 0.2s

```python
prob += hz1 >= a
prob += hz1 >= b
prob += hz1 >= 1/2
prob += hz1 <= a + 1000000*hzPrime1
prob += hz1 <= b + 1000000*hzPrime1
prob += hz1 <= 1/2 + 1000000*(1-hzPrime1)
```

✓ 0.1s

```
prob += p >= a + b + c
prob += p >= 1
prob += p <= a + b + c + 1000000*pPrime
prob += p <= 1 + 100000*(1-pPrime)
```
✓ 0.1s

Link to code : [Github]()

- Also tried different combination of constrains which could give optimal solutions to near 0.