Introduction to Linear Programming

Junaid Hasan

Math 381 Lecture 2

A sample problem

- Jack wants to buy oranges and apples.
- Suppose apples are priced at 3 dollars a pound and oranges at 4 dollars a pound.
- Each pound of apples has 70 grams of protein while each pound of orange has 50 grams of protein.
- A pound of apples has 50 grams of fiber while a pound of oranges has 80 grams of fiber.
- He wants to have at least 100 grams of fiber and at least 50 grams of protein.
- How much apples and oranges should he buy to minimize cost.

The Mathematical Model

- Suppose Jack buys x pounds of apples and y pounds of oranges.
- He wants to minimize $3x + 4y$.
- However he must ensure that $70x + 50y \geq 50$ (protein requirement) and
- that $50x + 80y \geq 60$ for the fiber requirement.

Discussion

- What do you notice here?

Solution by hand

- Lets plot something

(continued)..

My Plot

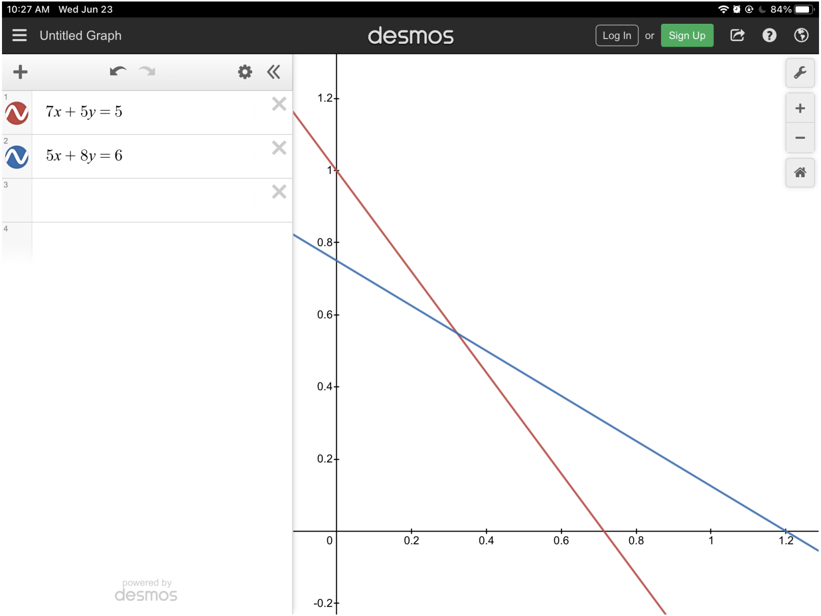


Figure 1: graph

(contd..)

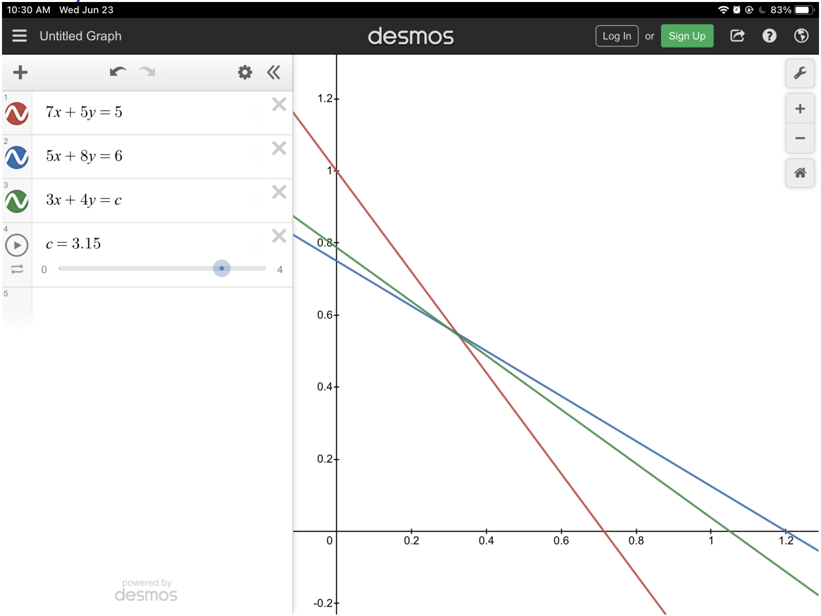


Figure 2: graph

Solution by code

- First we import the Model class from pyscipopt via `from pyscipopt import Model`
- Next Let us create a new object from the Model class and give it a name `model = Model("Apple and Oranges")`
- In the Model class we can add variables and constraints. Lets add the two variables
- `x = model.addVar(vtype="C", name="x")`
- `y = model.addVar(vtype="C", name="x")`
- Here the `vtype="C"` tells the program that the variables are continuous variables. If we had force the variables to be integers then we would use `vtype="I"`
- Lets add the constraints `model.addCons(70*x + 50*y >=50, "protein requirement")`
- `model.addCons(50*x + 80*y >=60, "fiber requirement")`

contd..

- Remark on `addVar()`. Its complete signature is
- `addVar(name="", vtype="C", lb=0.0, ub=None, obj=0.0, pricedVar = False)`
- Note the default values for names, vtype, lb, ub
- This means we could have called the function as `x = model.addVar("x")`
- Let us finally define the objective function `model.setObjective(3*x + 4*y, "minimize")`
- Solve by `model.optimize()`

Continued

- Input

```
from pyscipopt import Model
model = Model("Apple and Oranges")

x = model.addVar(vtype="C", name="x")
y = model.addVar(vtype="C", name="x")

model.addCons(70*x + 50*y >=50, "protein requirement")
model.addCons(50*x + 80*y >=60, "fiber requirement")

model.setObjective(3*x + 4*y, "minimize")

model.optimize()
```

contd..

```
if model.getStatus() == "optimal":
    print("Optimal value:", model.getObjVal())
    print("Solution:")
    print("  x = ", model.getVal(x))
    print("  y = ", model.getVal(y))
else:
    print("Problem could not be solved to optimaly")
```

- Output

```
Optimal value: 3.161290322580645
Solution:
  x =  0.3225806451612903
  y =  0.5483870967741935
```

Takeaways

- The problem has a linear (objective function) max/min objective.
- The constraints are linear as well.
- Implicit non-negativity constraints.
- Sometimes could be integer constraints (Integer LP).
- This kind of problem is known as a Linear Programming Problem.
- If you are interested in the history check out Wikipedia.

Standard Terminology

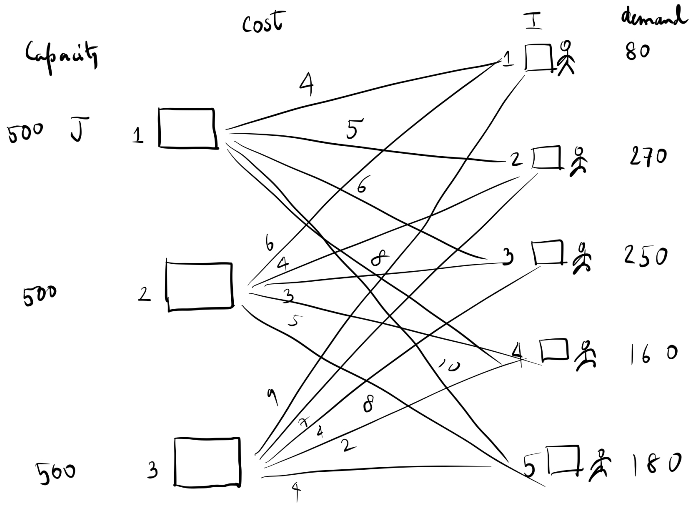
- A problem of the form

$$\begin{aligned} \text{maximize/minimize } & c_1x_1 + c_2x_2 \cdots + c_nx_n \\ \text{subject to } & a_{1,1}x_1 + \cdots + a_{1,n}x_n \leq \text{(or)} \geq \quad b_1 \\ & a_{2,1}x_1 + \cdots + a_{2,n}x_n \leq \text{(or)} \geq \quad b_2 \\ & \vdots \\ & a_{m,1}x_1 + \cdots + a_{m,n}x_n \leq \text{(or)} \geq \quad b_m \\ & x_1, x_2, \ldots, x_n \geq 0 \end{aligned}$$

- is called a standard linear programming problem.

Another Example: Transportation Problem

- A sports equipment company XYZ has products manufactured at three factories(j =1,2,3) and delivered to five stores(i = 1,2,3,4,5). What to do to minimize cost.
- Lets us draw the scenario



Transportation Problem contd.

- As a table it is given by

Transportation cost c_{ij}	Customer i: 1	2	3	4	5	Capacity
plant j: 1	4	5	6	8	10	500
2	6	4	3	5	8	500
3	9	7	4	2	4	500
demand d_i	80	270	250	160	180	

Problem Formulation

- Let x_{ij} be the amount of goods transported from factory j to customer i.
- Then can you write the optimization problem.

Answer:

- $$\begin{aligned} \text{minimize} \quad & \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} & \\ \text{subject to demand} \quad & \sum_{j \in J} x_{ij} = d_i & \forall i \in I \\ \text{and factory capacity} \quad & \sum_{i \in I} x_{ij} \leq M_j & \forall j \in J \\ & x_{ij} \geq 0 & \forall i \in I, j \in J \end{aligned}$$

Solution via code

- We will use a special datatype in Python to our advantage. It is a dictionary(or arrays would work fine as well). It allows to specify the keys and the values at the keys `{key1:value1, key2:value2, ...}`
- For demand `demand = {1:80, 2:270, 3:250, 4:160, 5:180}`
- For capacity `capacity = {1:500, 2:500, 3:500}`
- List(arrays in Python are called lists) of customers `I = [1,2,3,4,5]`
- Factories `J = [1,2,3]`
- Shipping cost would be a 2D array. We use dictionaries as `cost = {(1,1):4, (1,2):6, (1,3):9 (2,1):5, (2,2):4, (2,3):7, (3,1):6, (3,2):3, (3,3):3, (4,1):8, (4,2):5, (4,3):3, (5,1):10, (5,2):8, (5,3):4, }`

The full code and solution

(Refer to the python notebook attached in Canvas>Files)

- Output

```
Optimal value: 3350.0
sending quantity      80.0 from factory   1 to customer
sending quantity     270.0 from factory   2 to customer
sending quantity     230.0 from factory   2 to customer
sending quantity      20.0 from factory   3 to customer
sending quantity     160.0 from factory   3 to customer
sending quantity     180.0 from factory   3 to customer
```

Summary

- We saw problems that are known are linear programs.
- We say two examples.
- Some drawbacks of LPs
- The constraints and objective functions are linear. In nature we often have factors that are not linear say $ 5x^2 + y^2 $ or $ log(5x) + 4y $
- These do not work well in LPs.
- However the advantage of LPs is that they are simple to state and solve.
- Next class we will talk more about LPs and Integer LPs.