

Linear Programming Examples using Julia/JuMP

Antonio Flores-Tlacuahuac

Energy Research Group

Tecnológico de Monterrey, México

February 14, 2019

Example 1: Avocado Plant

An avocado oil company owns two A and B processing plants. Plants A and B can process a maximum of 460 and 650 Ton, respectively. The labor cost turns is 26 and 21 \$/Ton for plants A and B , also respectively. The avocado oil sale price turns out to be \$50/T regardless the manufacturing plant.

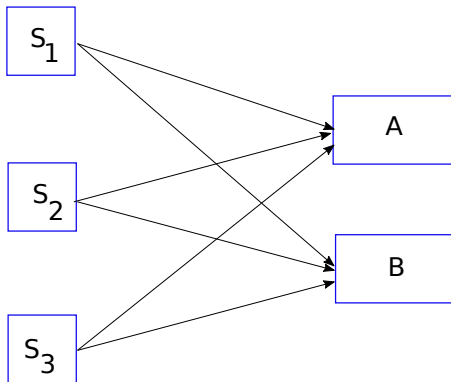
To meet the avocado oil manufacturing aims, the company can purchase avocado raw material from three different sellers: S_1 , S_2 and S_3 . In the following table, maximum avocado supply and purchase cost from each seller are shown.

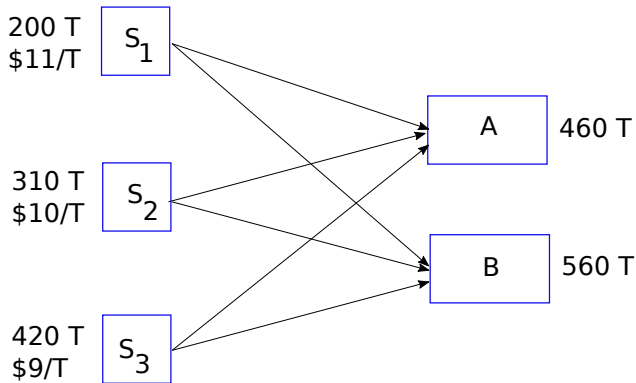
Seller	Maximum Supply [T]	Purchase cost [\$/T]
S_1	200	11
S_2	310	10
S_3	420	9

Moreover, transportation costs [\$/T] should also be taken into account to get a realistic net profit. Transportation cost are shown in next table.

Seller	Plant A	Plant B
S_1	3	3.5
S_2	2	2.5
S_3	6	4

Problem: We would like to compute the avocado amount to be purchase from each seller and the amount to be sent to each manufacturing plant such that net profit is maximized.





Results

- Amount [Ton] of avocado shipped to each plant

Seller	Plant A	Plant B
S_1	60	140
S_2	310	0
S_3	0	120

- Amount of avocado oil [Ton] manufactured at each plant

Plant	Amount [Ton]
A	370
B	560

- Economic Analysis

Gross Profit	46500
Labor Cost	21380
Raw Material Cost	9080
Transport Cost	2970.0
Net Profit	13070

► JuMP code

```
using JuMP, Clp
avocado_model = Model(solver=ClpSolver())
@variable(avocado_model, s1a >= 0, start = 100)
@variable(avocado_model, s2a >= 0, start = 150)
@variable(avocado_model, s3a >= 0, start = 200)
@variable(avocado_model, s1b >= 0, start = 100)
@variable(avocado_model, s2b >= 0, start = 150)
@variable(avocado_model, s3b >= 0, start = 200)
@variable(avocado_model, pa >= 0, start = 150)
@variable(avocado_model, pb >= 0, start = 200)
@objective(avocado_model, Max, 50pa+50pb-26pa-21pb-11(s1a+s1b)-10(s2a+s2b)
-9(s3a+s3b)-3s1a-3.5s1b- 2s2a -2.5s2b - 6s3a - 4s3b)
@constraint(avocado_model, c1, s1a + s1b <= 200)
@constraint(avocado_model, c2, s2a + s2b <= 310)
@constraint(avocado_model, c3, s3a + s3b <= 420)
@constraint(avocado_model, c4, s1a + s2a + s3a <= 460)
@constraint(avocado_model, c5, s1b + s2b + s3b <= 560)
@constraint(avocado_model, c6, s1a + s2a + s3a == pa)
@constraint(avocado_model, c7, s1b + s2b + s3b == pb)
print(avocado_model)
status_avocado_model = solve(avocado_model)
println("Status of solution:", status_avocado_model)
println("s1a =", getvalue(s1a))
println("s2a =", getvalue(s2a))
println("s3a =", getvalue(s3a))
println("s1b =", getvalue(s1b))
println("s2b =", getvalue(s2b))
println("s3b =", getvalue(s3b))
println("pa =", getvalue(pa))
println("pb =", getvalue(pb))
```



```
gross_profit = 50*getvalue(pa)+50*getvalue(pb)
labor = 26getvalue(pa)+21getvalue(pb)
raw_material = 11(getvalue(s1a)+getvalue(s1b))+10(getvalue(s2a)+getvalue(s2b))
+9(getvalue(s3a)+getvalue(s3b))
transport = 3getvalue(s1a)+3.5getvalue(s1b)+ 2getvalue(s2a) +2.5getvalue(s2b)
+ 6getvalue(s3a) + 4getvalue(s3b)
println("Gross Profit = ", gross_profit)
println("Labor cost = ", labor)
println("Raw Mat. cost = ", raw_material)
println("Transport cost = ", transport)
```

Example 2: Polymer Plant

A chemical company runs a polymer plant where three different grades A , B , C are manufactured. There are available 4 different batch reactors (denoted as R_1 , R_2 , R_3 and R_4) for polymers manufacture. Although the manufactured batch size for each grade is the same regardless of the type of reactor, each reactor requires different processing time [min/batch] for polymer production as shown in the next table.

Grade	R_1	R_2	R_3	R_4
A	5	7	4	10
B	6	12	8	15
C	13	14	9	17

The sale price [\$/batch] of each polymer is different depending on the reactor where it was manufactured as shown in the next table:

Grade	R_1	R_2	R_3	R_4
A	10	8	6	9
B	18	20	15	17
C	15	6	13	7

Polymer manufacture should meet the following grades demand:

Grade	Demand [number of batches]
A	100
B	150
C	100

How many batches of each grade should be manufactured such that polymer demands are met, assuming a 40 h/week working load, such that process profit is maximized?

R_1

A
B
C

R_2

A
B
C

R_3

A
B
C

R_4

A
B
C

Results

We have solved the problem under the following 2 operating scenarios:

- No Inventory is allowed

Profit = \$5500

Grade	R_1	R_2	R_3	R_4	Σ
A	100	0	0	0	100
B	0	150	0	0	150
C	100	0	0	0	100

- Some kind of inventory would be available

Profit = \$17882.5

Grade	R_1	R_2	R_3	R_4	Σ
A	0	0	100	0	100
B	400	200	137.5	160	897.5
C	0	0	100	0	100

► JuMP code

```
using JuMP, Clp
polymer_plant = Model(solver=ClpSolver())
@variable(polymer_plant, bar1 >= 0, start = 50)
@variable(polymer_plant, bar2 >= 0, start = 50)
@variable(polymer_plant, bar3 >= 0, start = 50)
@variable(polymer_plant, bar4 >= 0, start = 50)
@variable(polymer_plant, bbr1 >= 0, start = 50)
@variable(polymer_plant, bbr2 >= 0, start = 50)
@variable(polymer_plant, bbr3 >= 0, start = 50)
@variable(polymer_plant, bbr4 >= 0, start = 50)
@variable(polymer_plant, bcr1 >= 0, start = 50)
@variable(polymer_plant, bcr2 >= 0, start = 50)
@variable(polymer_plant, bcr3 >= 0, start = 50)
@variable(polymer_plant, bcr4 >= 0, start = 50)
@objective(polymer_plant, Max, 10bar1+8bar2+6bar3+9bar4+18bbr1+20bbr2+15bbr3
+17bbr4+15bcr1+6bcr2+13bcr3+7bcr4)
@constraint(polymer_plant, c1, bar1 + bar2 + bar3 + bar4 >= 100)
@constraint(polymer_plant, c2, bbr1 + bbr2 + bbr3 + bbr4 >= 150)
@constraint(polymer_plant, c3, bcr1 + bcr2 + bcr3 + bcr4 >= 100)
@constraint(polymer_plant, c4, 5bar1 + 6bbr1 + 13bcr1 <= 2400)
@constraint(polymer_plant, c5, 7bar2 + 12bbr2 + 14bcr2 <= 2400)
@constraint(polymer_plant, c6, 4bar3 + 8bbr3 + 9bcr3 <= 2400)
@constraint(polymer_plant, c7, 10bar4 + 15bbr4 + 17bcr4 <= 2400)
print(polymer_plant)
status_polymer_model = solve(polymer_plant)
```

► JuMP code

```
println(" Objective function value = ", getobjectivevalue(polymer_plant))
println(" bar1 =", getvalue(bar1))
println(" bar2 =", getvalue(bar2))
println(" bar3 =", getvalue(bar3))
println(" bar4 =", getvalue(bar4))
println(" bbr1 =", getvalue(bbr1))
println(" bbr2 =", getvalue(bbr2))
println(" bbr3 =", getvalue(bbr3))
println(" bbr4 =", getvalue(bbr4))
println(" bcr1 =", getvalue(bcr1))
println(" bcr2 =", getvalue(bcr2))
println(" bcr3 =", getvalue(bcr3))
println(" bcr4 =", getvalue(bcr4))
```

Optimal Power Contract Capacity

Commonly, power generating enterprises have different tariffs depending upon the type of consumers. Hence, consumers are divided as follows:

- ▶ Residential
- ▶ Commercial
- ▶ Industrial

Moreover, during a typical day power supply is split in fixed power periods:

- ▶ Peak
- ▶ Medium
- ▶ Off-Peak

Among other factors, power tariff depends upon the above power supply periods.

Optimal Power Contract Capacity

Due to its economic impact , we will only address the optimal negotiating of a one year Industrial power contract.

The tariff of an Industrial power contract is split into two parts:

$$\text{Cost} = \text{Energy Charge} + \text{Capacity Charge}$$

- ▶ Energy Charge: This charge is fixed and depends only on power consumption and power supply period
- ▶ Capacity Charge: Commonly, this is the term that we should negotiate in an optimal manner. Due to its importance we we will address it in a separate way.

Optimal Power Contract Capacity

Some Capacity Charge Considerations...

- ▶ Determined by user consumption [kW/month] based on maximum demand during time of use
- ▶ If power demand in Peak period does not exceed the nominal contract capacity then a fixed capacity charge is levied
- ▶ If power demand in Peak period exceeds nominal contract capacity then a penalty charge (i.e. 2-3 times power basic cost) applies
- ▶ Hence, choosing a low power capacity contract likely will enforce high capacity charge
- ▶ On the other hand, choosing a high power contract capacity may result in unnecessary capacity charge

Optimal Power Contract Capacity

Constraints

- ▶ **Energy charge:** Power cost during a given consumption period
- ▶ **Capacity charge:** (a) If power consumer demand exceeds the nominal capacity contract within 10% of the nominal contract capacity then a penalty is charged at twice the rate of nominal contract capacity and (b) Excess power demand over 10% of the nominal contract capacity is charged three times that rate
- ▶ **Power factor adjustment:** (a) Monthly bill will be reduced by 0.15% for every 1% of the average monthly power factor above 80%, (b) It will be increased by 0.3% for every 1% below 80%

Optimal Power Contract Capacity

Constraints

- ▶ **Expanding construction fee:** Power consumers may request a change in their peak contract capacity each month. However, an expanding construction fee will be charged. The expanding construction rate is 1759 [\$/kW] for summer months and 1320 for non-summer months.
- ▶ **Disallowed decreased in contract capacities:** At end of each month customers can ask for a decrease in the power contract capacity. However, if they ask for an increase in power capacity within two years they are forced to pay a maintenance bill that will cost more than staying with the original power contract capacity. Therefore, avoid this scenario.

Optimal Power Contract Capacity

- ▶ Power factor = 98%
- ▶ Peak period power cost (summer months) = 224 [\$/kW]
- ▶ Peak period power cost (non-summer months) = 167 [\$/kW]

Month	Maximum Demand [kWh]	Peak Contract Capacity [kWh]
1	5048	4950
2	5144	4950
3	4616	4950
4	4296	4950
5	3696	4950
6	3392	4950
7	5048	4950
8	5144	4950
9	4616	4950
10	4296	4950
11	3696	4950
12	3392	4950