Supply Chain Network Design to Support Biofuel Production: A case study

Solution to the case study model designed by **Vaibhav Budhkar**

Parameters

1. inv_hub = 3476219 (investment cost of the hub)
2. inv_ref = 130956797 (investment cost of the bio-refinery)
3. cost_train = 3066792 (train loading/unloading cost)
4. load_train = 338000 (train loading/unloading capacity)
5. out_scr = 1000 (outsourcing cost per Mg)
6. conv_yld = 232 (conversion yield liters/Mg)
7. demand_tot = 1476310602 (Total demand in liters)
8. hub_cap = 300000 (hub capacity)
9. plant_cap = 655447.004 (Plant capacity)
10. s1 = total number of counties
11. s2 = total number of hubs
12. s3 = total number of bio-refinery
13. cost_ch = cost of biomass/ Mg from county to hub
14. cost_hp = cost of biomass/ Mg from hub to county

Decision Variables

X_{ij} = amount in Mg shipped from county to hub

Y_{jk} = amount in Mg shipped from hub to bio-refinery

P_j = binary variable (= 1 ; if hub is open || = 0 otherwise)

q_k = binary variable (=1 ; if refinery is open || = 0 otherwise)

r_{jk} = binary variable (= 1 ; if railway track is used || = 0 otherwise)

h = imported supply

Objective function

Min Z = inv_hub* $\sum_{j=1}^{1303}$ $\mathbf{p_j}$ + inv_ref* $\sum_{k=1}^{167}$ $\mathbf{q_k}$ +

$\sum_i \sum_j (\mathbf{x_{ij}} * \text{cost\_ch}_{ij}) + \sum_j \sum_k (\mathbf{y_{jk}}\ \text{cost\_hp}_{jk} + \text{cost\_train}* \mathbf{r_{jk}}) + \text{out\_scr}*\mathbf{h}$

Subject to Constraints:

Supply capacity

1. $\sum_j \mathbf{x[i,j]} \leq supplier\ capacity[i]$, for ∀ i ϵ{1,2,……, no. of suppliers}, where j ϵ {1,2,……, no. of hubs}

Hub capacity

2. $\sum_i \mathbf{x[i,j]} \leq 300000 * \mathbf{p[j]}$, for ∀ j ϵ{1,2,……, no. of hubs}, where i ϵ{1,2,……, no. of suppliers}

Bio-refinery Capacity

3. $\sum_j \mathbf{y[j,k]} \leq 655447.004 * \boldsymbol{q}[\mathbf{k}]$, for ∀ k ϵ{1,2,……, no. of bio-refinery}, where j ϵ {1,2,……, no. of hubs}

Demand constraint

4. $\sum_j \sum_k \mathbf{y[j,k]} + \boldsymbol{h} = \frac{demand\ \_tot}{conv\ \_yld} Mg$

Flow balance

5. $\sum_i \mathbf{x[i,j]} = \sum_k \mathbf{y[j,k]}$ for ∀ j ϵ{1,2,……, no. of hubs}, where i ϵ{1,2,……, no. of suppliers} and k ϵ{1,2,……,no. of biorefinery}

Train Capacity

6. $\mathbf{Y_{j,k}} \leq 338000 * \mathbf{r_{[j,k]}}$, for ∀ j ϵ{1,2,……, no. of hubs}, k ϵ{1,2,……,no. of biorefinery}

Julia Code:

```
using JuMP, Gurobi, CSVFiles, DataFrames, CSV
m = Model(solver = GurobiSolver(MIPGap=0.05)) #model definition

dir1 = "C:/Users/pbudh/Desktop/CLemson Sem_01/Engineering Optimization and
Application/Extra Credit Work/TX_suppliers.csv"
dir1_data = CSV.read(dir1)
s1 = nrow(dir1_data)

dir2 = "C:/Users/pbudh/Desktop/CLemson Sem_01/Engineering Optimization and
Application/Extra Credit Work/TX_hubs.csv"
dir2_data = CSV.read(dir2)
s2 = nrow(dir2_data)

dir3 = "C:/Users/pbudh/Desktop/CLemson Sem_01/Engineering Optimization and
Application/Extra Credit Work/TX_plants.csv"
dir3_data = CSV.read(dir3)
s3 = nrow(dir3_data)

# Parameters-Definition

loc1 = "C:/Users/pbudh/Desktop/CLemson Sem_01/Engineering Optimization and
Application/Extra Credit Work/TX_roads.csv"
loc_data1 = DataFrame(load(loc1)) #stores the roads.csv file in "loc_data1"
cost_ch= permutedims(reshape(loc_data1[1:end, :cost], s1, s2), (1,2)) #reshape function
returns matrix and stores it in "cost_ch"
loc2 = "C:/Users/pbudh/Desktop/CLemson Sem_01/Engineering Optimization and
Application/Extra Credit Work/TX_railroads.csv"
loc_data2 = DataFrame(load(loc2))
cost_hp= permutedims(reshape(loc_data2[1:end, :cost], s2, s3), (1,2))
inv_hub = 3476219 #investment cost of the hub
inv_ref = 130956797 #investment cost of the bio-refinery
cost_train = 3066792 #train loading/unloading cost
load_train = 338000 #train loading/unloading capacity
out_scr = 1000 #outsourcing cost per Mg
conv_yld = 232 #conversion yeild liters/Mg
demand_tot = 1476310602 #Total demand in liters
hub_cap = 300000 #hub capacity
plant_cap = 655447.004 #plant capacity

# End of Parameters-Definition

# Decision Variable Declaration

@variable(m, x[1:s1,1:s2]>=0) #Xij,mass from county to hub
@variable(m, y[1:s2,1:s3]>=0) #Yjk mass from hub to biorefinery
```

```
@variable(m, p[1:s2], Bin) #Pj is the binary variable if hub is open
@variable(m, q[1:s3], Bin) #Qk is the binary variable if refinery is open
@variable(m, r[1:s2,1:s3], Bin) #Rjk is the binary variable for rail tracks
@variable(m, h>=0) #imported supply h value from 0 to demand dk=6363408

#End of Decision Variable Declaration

#Objective function

@objective(m, Min, sum(x[i,j]*cost_ch[i,j] for i=1:s1,j=1:s2) + sum(y[j,k]*cost_hp[j,k] for
j=1:s2,k=1:s3) + sum(cost_train*r[j,k] for j=1:s2,k=1:s3) + sum(p[j] for j=1:s2)*inv_hub +
sum(q[k] for k = 1:s3)*inv_ref + out_scr*h)

#End of objective function

#Constraints Definition

@constraint(m, [i = 1:s1], sum(x[i,j] for j = 1:s2)<= dir1_data[i,:Supply]) #Supply capacity
constraint
@constraint(m, [j = 1:s2], sum(x[i,j] for i = 1:s1)<= hub_cap*p[j]) #hub capacity constraint
@constraint(m, [k = 1:s3], sum(y[j,k] for j = 1:s2)<= plant_cap*q[k]) #plant capacity constraint
@constraint(m, sum(sum(y[j,k] for k = 1:s3) for j = 1:s2) + h == demand_tot/conv_yld) #demand
constraint
@constraint(m, [j = 1:s2], sum(x[i,j] for i = 1:s1) == sum(y[j,k] for k = 1:s3)) #flow balance
constraint
@constraint(m, [j = 1:s2, k = 1:s3], y[j,k] <= load_train*r[j,k]) #train constraint

#end of constraint definition

status = solve(m)
hubs = getvalue(p)
for j in 1:s2
      if hubs[j]==1
      println("hubs[$j]=",hubs[j])
      else
      end
      end
plants = getvalue(q)
for k in 1:s3
   if plants[k]==1
   println("plants[$k]=",plants[k])
else
end
end
rail_tracks = getvalue(r)
for j in 1:s2
   for k in 1:s3
   if rail_tracks[j,k] == 1
      println("rail_tracks[$j,$k]",rail_tracks[j,k])
   else
   end
```

```
    end
end
outsourced_mass = getvalue(h)
println("biomass outsourced is ", outsourced_mass)
zcost = getobjectivevalue(m)

#Unit cost of Production

tot_supply = sum(dir1_data[1:end, :Supply])
unit_cost_prod1 = (zcost - outsourced_mass*out_scr)/tot_supply
```

Julia Output and Results:

- Output for solving the model.

```
julia> status = solve(m)
Academic license - for non-commercial use only
Optimize a model with 220629 rows, 767635 columns and 2082362 nonzeros
Variable types: 548564 continuous, 219071 integer (219071 binary)
Coefficient statistics:
  Matrix range     [1e+00, 7e+05]
  Objective range  [2e+00, 2e+11]
  Bounds range     [1e+00, 1e+00]
  RHS range        [8e-01, 6e+06]
Warning: Model contains large objective coefficients
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Found heuristic solution: objective 6.363408e+09
Presolve removed 6 rows and 7818 columns (presolve time = 6s) ...
Presolve removed 6 rows and 7818 columns
Presolve time: 7.18s
Presolved: 220623 rows, 759817 columns, 2058908 nonzeros
Variable types: 540746 continuous, 219071 integer (219071 binary)

Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.5045646e+07   1.557923e+06   0.000000e+00     10s
     379    1.5861033e+07   8.047474e+05   0.000000e+00     10s
    8013    1.5560100e+08   2.350930e+06   0.000000e+00     15s
   13426    1.6709872e+08   1.627888e+06   0.000000e+00     20s
   18122    1.7484770e+08   2.360572e+06   0.000000e+00     25s
   23319    1.8209128e+08   2.360572e+06   0.000000e+00     30s
   27929    1.8770801e+08   1.597434e+06   0.000000e+00     35s
   32801    1.9446019e+08   1.884453e+06   0.000000e+00     40s
   38932    1.4557813e+09   4.329527e+06   0.000000e+00     45s
   41760    1.4626020e+09   1.939195e+06   0.000000e+00     50s
   43960    1.4650532e+09   1.525132e+06   0.000000e+00     55s
   45660    1.4666738e+09   2.585254e+06   0.000000e+00     61s
   47300    1.4682352e+09   1.740235e+06   0.000000e+00     66s
   48770    1.4695679e+09   1.890453e+06   0.000000e+00     70s
   50200    1.4707718e+09   8.731481e+05   0.000000e+00     75s
   51630    1.4720756e+09   8.419746e+05   0.000000e+00     81s
   52870    1.4733723e+09   1.517051e+06   0.000000e+00     85s
   54580    1.4756671e+09   1.253084e+06   0.000000e+00     91s
```

- Objective function value = **4.014981 * 10^9** (with a gap of 5%)

```
Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.5045646e+07   1.557923e+06   0.000000e+00     10s
     379    1.5861033e+07   8.047474e+05   0.000000e+00     10s
    8013    1.5560100e+08   2.350930e+06   0.000000e+00     15s
   13426    1.6709872e+08   1.627888e+06   0.000000e+00     20s
   18122    1.7484770e+08   2.360572e+06   0.000000e+00     25s
   23319    1.8209128e+08   2.360572e+06   0.000000e+00     30s
   27929    1.8770801e+08   1.597434e+06   0.000000e+00     35s
   32801    1.9446019e+08   1.884453e+06   0.000000e+00     40s
   38932    1.4557813e+09   4.329527e+06   0.000000e+00     45s
   41760    1.4626020e+09   1.939195e+06   0.000000e+00     50s
   43960    1.4650532e+09   1.525132e+06   0.000000e+00     55s
   45660    1.4666738e+09   2.585254e+06   0.000000e+00     61s
   47300    1.4682352e+09   1.740235e+06   0.000000e+00     66s
   48770    1.4695679e+09   1.890453e+06   0.000000e+00     70s
   50200    1.4707718e+09   8.731481e+05   0.000000e+00     75s
   51630    1.4720756e+09   8.419746e+05   0.000000e+00     81s
   52870    1.4733723e+09   1.517051e+06   0.000000e+00     85s
   54580    1.4756671e+09   1.253084e+06   0.000000e+00     91s
   56180    1.4800599e+09   9.531259e+05   0.000000e+00     95s
   58388    4.0149810e+09   0.000000e+00   0.000000e+00     98s

Root relaxation: objective 4.014981e+09, 58388 iterations, 89.37 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 4.0150e+09    0  375 6.3634e+09 4.0150e+09  36.9%     -  102s
H    0     0                     6.208117e+09 4.0150e+09  35.3%     -  102s
H    0     0                     4.090593e+09 4.0150e+09  1.85%     -  147s

Explored 1 nodes (58388 simplex iterations) in 148.07 seconds
Thread count was 4 (of 4 available processors)

Solution count 3: 4.09059e+09 6.20812e+09 6.36341e+09

Optimal solution found (tolerance 5.00e-02)
Best objective 4.090592697476e+09, best bound 4.014980958977e+09, gap 1.8484%
:Optimal
```

- Number of hubs that are built to preprocess and reconsolidate = **11**

```
julia> for j in 1:s2
            if hubs[j]==1
                  println("hubs[$j]=",hubs[j])
                        else
                              end
                                    end
hubs[99]=1.0
hubs[218]=1.0
hubs[288]=1.0
hubs[292]=1.0
hubs[428]=1.0
hubs[515]=1.0
hubs[516]=1.0
hubs[643]=1.0
hubs[810]=1.0
hubs[928]=1.0
hubs[929]=1.0
```

- Number of bio-refineries that are built to convert the biomass to bio-fuel = **5**

```
julia> for k in 1:s3
          if plants[k]==1
              println("plants[$k]=",plants[k])
              else
              end
              end
plants[37]=1.0
plants[61]=1.0
plants[108]=1.0
plants[153]=1.0
plants[156]=1.0
```

- Amount of mass in Mg that is imported to meet the demand = **3.3100300589 * 10^6**

```
julia> outsourced_mass = getvalue(h)
3.310030058978753e6

julia> println("biomass outsourced is ", outsourced_mass)
biomass outsourced is 3.310030058978753e6
```

- Number of train tracks used = **11**

```
julia> for j in 1:s2
           for k in 1:s3
               if rail_tracks[j,k] == 1
                       println("rail_tracks[$j,$k]",rail_tracks[j,k])
                           else
                               end
                                   end
                                   end
rail_tracks[99,108]1.0
rail_tracks[218,156]1.0
rail_tracks[288,108]1.0
rail_tracks[292,156]1.0
rail_tracks[428,37]1.0
rail_tracks[515,61]1.0
rail_tracks[516,61]1.0
rail_tracks[643,153]1.0
rail_tracks[810,153]1.0
rail_tracks[928,37]1.0
rail_tracks[929,37]1.0
```

- Unit cost of production = **$255.63**

```
julia> unit_cost_prod1 = (zcost - outsourced_mass*out_scr)/tot_supply
255.63907026135124
```