Chair Logistics 2

The Chair Logistics 2 assignment is an extension of the Chair Logistics assignment, now allowing establishment and closing of depots. A depot either is used or not, a binary decision, hence binary variables are needed. Here we give the full model, but only one variable $q_d \in \{0, 1\}$ is added and then used in the objective function and set in one constraint. Also, the number of **possible** depots is extended to 6.

Problem

- Minimize the total truck transportation costs.

Sets

- $p \in Plants = \{P1, P2\}$
- $d \in Depots = \{D1, D2, D3, D4\}$
- $r \in Retailer = \{R1, R2, R3, R4, R5, R6\}$

Parameters

- $PlantCapacity_p$: Maximal chair production capacity for plant p
- $DepotCapacity_d$: Chair capacity at depot d
- $RetailerRequirements_r$: Chair demand at retailer r
- F: The cost of transporting one chair one kilometer
- $PDdist_{p,d}$: Distance in kilometers from plant p to depot d
- $PRdist_{p,r}$: Distance in kilometers from plant p to retailer r
- $DRdist_{p,d}$: Distance in kilometers from depot d to retailer r
- $TruckCostKM = 1.5$: Cost in € pr. truck pr. kilometer
- $TruckCap = 40$: Number of chairs which can be transported on a truck

- $DepotConstructCost_d$: The construction cost of depot d, 0 for existing depots.
- $DepotCancelProfit_d$: The savings of closing depot d, 0 for new depots.

Decision variables

- Amount of chairs transported from plant p to depot d: $xpd_{p,d} \geq 0$.
- Amount of chairs transported from plant p to retailer r: $xpr_{p,r} \geq 0$.
- Amount of chairs transported from depot d to retailer r: $xdr_{d,r} \geq 0$.
- Number of trucks transporting from plant p to depot d: $ypd_{p,d} \in Z^+$.
- Number of trucks transporting from plant p to retailer r: $ypr_{p,r} \in Z^+$.
- Number of trucks transporting from depot d to retailer r: $ydr_{d,r} \in Z^+$.
- Is depot d used or not: $q_{d,} \in \{0, 1\}$.

Model

Objective:

- Total truck transport costs minimized:

$$\sum_{p,d} TruckCostKM \cdot PDdist_{p,d} \cdot ypd_{p,d} +$$

$$\sum_{p,r} TruckCostKM \cdot PRdist_{p,r} \cdot ypr_{p,r} +$$

$$\sum_{d,r} TruckCostKM \cdot DRdist_{d,r} \cdot ydr_{d,r} +$$

$$\sum_{d} DepotConstructCost_d \cdot q_d -$$

$$\sum_{d} DepotCancelProfit_d \cdot (1 - q_d)$$

Constraints:

- Production capacity limit for plant p:

$$\sum_{d} xpd_{p,d} \leq PlantCapacity_p \; \forall \; p$$

- Chair capacity limit in depot d:

$$\sum_p xpd_{p,d} \leq DepotCapacity_d \cdot q_d \ \forall \ d$$

- What goes out of a depot d must come into the depot d:

$$\sum_p xpd_{p,d} = \sum_r xpr_{p,r} \ \forall \ d$$

- Ensure that chair demand at retailer r is satisfied:

$$\sum_p xpd_{p,r} + \sum_d xdr_{d,r} \geq RetailerRequirements_r \ \forall \ r$$

- Force the number of necessary trucks from plant p to depot d:

$$xpd_{p,d} \leq TruckCap \cdot ypd_{p,d} \ \forall \ p, d$$

- Force the number of necessary trucks from plant p to retailer r:

$$xpr_{p,r} \leq TruckCap \cdot ypd_{p,r} \ \forall \ p, r$$

- Force the number of necessary trucks from depot d to retailer r:

$$xpd_{d,r} \leq TruckCap \cdot ypd_{d,r} \ \forall \ d, r$$

Notice that the q_d variable is used to both paying for new constructions **and** subtract the savings. This is achieved by the **not** statement $(1-q_d)$. Notice also how we modify the depot capacity constraint to force the setting of q_d variable. Since it costs money to open a depot, only forcing opening a depot, i.e. setting it to 1 is necessary.

The full model in Julia/JuMP, available with the name

`ChairLogistics2.jl`

from the book web-site, is given below:

```
#*************************************************************************
# Chair Logistics Assignment, "Mathematical Programming Modelling" (42112)
using JuMP
using HiGHS
#*************************************************************************

#*************************************************************************
# PARAMETERS
```

```
Plants = ["P1", "P2"]            # set of plants
P = length(Plants) # P is the size of the Plants set
Depots = ["D1", "D2", "D3", "D4", "D5", "D6"]
D = length(Depots)
Retailers = ["R1", "R2", "R3", "R4","R5","R6"]
R = length(Retailers)
PlantCapacity = [7500, 8500] # plant capacity in chairs
DepotCapacity = [3250, 3500, 3500, 3000, 3750, 3250] # depot capacity in chairs
Ret = [1500 2500 2000 3000 2000 3000] # retailer capacity in chairs
PD_Dist = [ 137  92  48 173  88 109; # Distance between Plants and Depots
             54  109 111  85 128 105]

PR_Dist = [307 260 215 196 148 268;
           234 173 194 264 204 218]

DR_Dist = [ 109  58  65 187 128  88;
             214 163  54  89  26 114;
             223 173  97  71  29 162;
             81   51 133 239 170 155;
             223 172  62  80  13 124;
             185 135  31 113  54  96
             ]
TruckCap=40
TruckCostKM=1.5
DepotConstructCost=[   0    0    0    0 5000 5000]
DepotCancelProfit= [5000 5000 5000 5000    0    0]
#*****************************************************************************

#*****************************************************************************
# Model
CL = Model(HiGHS.Optimizer)

@variable(CL, xpd[p=1:P, d=1:D] >= 0)
@variable(CL, xpr[p=1:P, r=1:R] >= 0)
@variable(CL, xdr[d=1:D, r=1:R] >= 0)
@variable(CL, ypd[p=1:P, d=1:D] >= 0, Int)
@variable(CL, ypr[p=1:P, r=1:R] >= 0, Int)
@variable(CL, ydr[d=1:D, r=1:R] >= 0, Int)

@variable(CL, q[d=1:D], Bin) # 1 if depot exists

# Minimize transporatation cost
@objective(CL, Min,
           sum( TruckCostKM*PD_Dist[p,d]*ypd[p,d] for p=1:P, d=1:D) +
           sum( TruckCostKM*PR_Dist[p,r]*ypr[p,r] for p=1:P, r=1:R) +
           sum( TruckCostKM*DR_Dist[d,r]*ydr[d,r] for d=1:D, r=1:R) +
```

```
            sum( DepotConstructCost[d]*q[d] for d=1:D) -
            sum( DepotCancelProfit[d]*(1-q[d]) for d=1:D)
            )

@constraint(CL, [p=1:P],
             sum(xpd[p,d] for d=1:D) + sum(xpr[p,r] for r=1:R) <= PlantCapacity[p])

@constraint(CL, [d=1:D],
             sum(xdr[d,r] for r=1:R) <= DepotCapacity[d]*q[d])

@constraint(CL, [d=1:D],
             sum(xpd[p,d] for p=1:P) == sum(xdr[d,r] for r=1:R))

@constraint(CL, [r=1:R],
             sum(xpr[p,r] for p=1:P) + sum(xdr[d,r] for d=1:D) == Ret[r])


@constraint(CL,[p=1:P, d=1:D], xpd[p,d] <= TruckCap*ypd[p,d] )

@constraint(CL,[p=1:P, r=1:R], xpr[p,r] <= TruckCap*ypr[p,r] )

@constraint(CL,[d=1:D, r=1:R], xdr[d,r] <= TruckCap*ydr[d,r] )
#*****************************************************************************


#*****************************************************************************
# solve
optimize!(CL)
println("Termination status: $(termination_status(CL))")
#*****************************************************************************


#*****************************************************************************
# Report results
let
println("-------------------------------------");
if termination_status(CL) == MOI.OPTIMAL
    println("RESULTS:")
    println("objective = $(objective_value(CL))")
    for d=1:D
        if round.(Int8,value(q[d]))==1
            println("Using Depot: $(Depots[d])")
        else
            println("NOT Using Depot: $(Depots[d])")
        end
    end
```

```
else
  println("  No solution")
end
println("---------------------------------------");
end
#*****************************************************************************
```