Chair Distribution

The chair transport problem is now extended, such that transport directly to the end-customers, the retailers is included. The problem is now significantly larger, hence the full problem is againg stated.

Problem

- Minimize the total 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

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: $xdr_{d,r} \geq 0$.

Model

Objective:

- Total transport costs minimized:

$$\sum_{p,d} F \cdot PDdist_{p,d} \cdot xpd_{p,d} +$$

$$\sum_{p,r} F \cdot PRdist_{p,r} \cdot xpr_{p,r} +$$

$$\sum_{d,r} F \cdot DRdist_{d,r} \cdot xdr_{d,r}$$

Constraints:

- Production capacity limit for plant p:

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

- Chair capacity limit in depot d:

$$\sum_{p} xpd_{p,d} \leq DepotCapacity_d \ foralld$$

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

$$\sum_{p} xpd_{p,d} = \sum_{r} xpr_{p,r} \ \ foralld$$

- Ensure that chair demand at retailer r is satisfied:

$$\sum_{p} xpd_{p,r} + \sum_{d} xdr_{d,r} \geq RetailerRequirements_r \ \ forallr$$

Notice that in the above model, we use the correct indexes in the name of the variable, to make it easier to spot index errors. Notice also the balance constraint which ensures that what comes into a depot must also go out.

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

`ChairDistribution.jl`

from the book web-site, is given below:

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

#*************************************************************************
# PARAMETERS
Plants = ["P1", "P2"]             # set of plants
P = length(Plants) # P is the size of the Plants set
Depots = ["D1", "D2", "D3", "D4"]
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] # depot capacity in chairs
RetailerRequirements = [1500 2500 2000 3000 2000 3000] # retailer capacity in chairs
PDdist = [ 137  92  48 173; # Distance between Plants and Depots
            54  109 111  85]

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

DRdist = [ 109  58  65 187 128  88;
            214 163  54  89  26 114;
            223 173  97  71  29 162;
            81   51 133 239 170 155;
            ]
F=0.0375
#*************************************************************************

#*************************************************************************
# Model
CD = Model(HiGHS.Optimizer)
```

```
@variable(CD, xpd[p=1:P, d=1:D] >= 0)
@variable(CD, xpr[p=1:P, r=1:R] >= 0)
@variable(CD, xdr[d=1:D, r=1:R] >= 0)

# Minimize transporatation cost
@objective(CD, Min,
           sum( PDdist[p,d]*F*xpd[p,d] for p=1:P, d=1:D) +
           sum( PRdist[p,r]*F*xpr[p,r] for p=1:P, r=1:R) +
           sum( DRdist[d,r]*F*xdr[d,r] for d=1:D, r=1:R))

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

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

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

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

print(CD)
#**************************************************************************


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


#**************************************************************************
# Report results
let
println("------------------------------------");
if termination_status(CD) == MOI.OPTIMAL
    println("RESULTS:")
    println("objective = $(objective_value(CD))")


    @printf "Plant transports\n"
    for p=1:P
        @printf "\t%s\n" Plants[p]
        @printf "\t\tDepots\n"
```

```
        for d=1:D
            if value(xpd[p,d])>0.001
                @printf "\t\t%s\tTransport: %.2f\n" Depots[d] value(xpd[p,d])
            end
        end

        @printf "\t\tRetailers\n"
        for r=1:R
            if value(xpr[p,r])>0.001
                @printf "\t\t%s\tTransport: %.2f\n" Retailers[r] value(xpr[p,r])
            end
        end
        @printf "\n"
    end

    @printf "Depots\n"
    for d=1:D
        @printf "\t%s\n" Depots[d]
        for r=1:R
            if value(xdr[d,r])>0.001
                @printf "\t\t%s\tTransport: %.2f\n" Retailers[r] value(xdr[d,r])
            end
        end

        @printf "\n"
    end
else
  println("  No solution")
end
println("-------------------------------------");
end
#*****************************************************************************
```