Scrap Removal

In the scrap removal assignment you minimize the cost of removing the scrap, to earn money yourself.

Problem

- Minimize the scrap removal costs.

Sets

- Scrap items: $i \in Items = \{"S1","S2","S3","S4","S5","S6","S7","S8","S9","S10","S11","S12","S13","S14","S15","S16","S17","S18","S19","S20"\}$
- $b \in BigBags = \{12345678910\}$

Parameters

- $ItemWeight_i$: Weight of item i
- $BigBagCost = 50$: Cost for removing one bigbag
- $BigBagWeightLimit = 100$: How many kilo's can a bigbag hold

Decision variables

- Decide if scrap item i is put bib bigbag b: $x_{i,b} \in \{0,1\}$.
- Decide if you need to use bigbag b: $y_b \in \{0,1\}$

Model

Objective:

- Total cost of scrap removal:

$$\sum_b BigBagCost \cdot y_i$$

Constraints:

- Limit to bigbag capaicty **and** set the y variable:

$$\sum_i ItemWeight_i \cdot x_{i,b} \leq BigBagWeightLimit \cdot y_i \ \forall \ b$$

- Ensure that all the scrap is removed:

$$\sum_b x_{i,b} = 1 \ \forall \ i$$

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

`ScrapRemoval.jl`

from the book web-site, is given below:

```
#*****************************************************************************
# Scrap Removal Assignment, "Mathematical Programming Modelling" (42112)
using JuMP
using HiGHS
#*****************************************************************************

#*****************************************************************************
# PARAMETERS
Items = ["S1","S2","S3","S4","S5","S6","S7","S8","S9","S10",
         "S11","S12","S13","S14","S15","S16","S17","S18","S19","S20"]
I=length(Items)
ItemWeight=[35,10,45,53,37,22,26,38,63,17,44,54,62,42,39,51,24,52,46,29]
BigBags=[1,2,3,4,5,6,7,8,9,10]
B=length(BigBags)
BigBagCost=50
BigBagWeightLimit=100
#*****************************************************************************

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

@variable(SR, x[i=1:I,b=1:B], Bin) # x[i,b]=1 if item i is in big-bag b
```

```
@variable(SR, y[b=1:B], Bin)

# Minimize transporatation cost
@objective(SR, Min,
           sum( BigBagCost*y[b] for b=1:B)
           )

@constraint(SR, [b=1:B],
            sum( ItemWeight[i]*x[i,b] for i=1:I) <= BigBagWeightLimit*y[b]
            )

@constraint(SR, [i=1:I],
            sum( x[i,b] for b=1:B) == 1
            )
#**************************************************************************


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


#**************************************************************************
# Report results
let
println("---------------------------------------");
if termination_status(SR) == MOI.OPTIMAL
    println("RESULTS:")
    println("objective = $(objective_value(SR))")
else
  println("  No solution")
end
println("---------------------------------------");
end
#**************************************************************************
```