Stamps

You receive 213 bids for your 100 stamps. You want to earn as much money as possible

Problem

- Maximize the profit obtained by selling your stamps

Sets

- $s \in Stamps = \{1, 2, 3, ..., 211, 212, 213\}$
- $b \in Bids == \{1, 2, 3, ..., 98, 99, 100\}$

Parameters

- $BidPrice_b$: The price the buyer will pay for all the stamps in this bid
- $BidSets_{b,s}$: Incidence matrix where $BidSets[b, s] = 1$ if bid b includes stamp s and is otherwise 0

Decision variables

- Should you accept bid b: $x_b \in \{0, 1\}$

Model

Objective:

- Maximize sales profit:

$$\sum_b BidPrice_b \cdot x_b$$

Constraints:

- You can only sell each stamp once:

$$\sum_b BidSets_{b,s} \cdot x_b \leq \ \forall\ s$$

The above model is the classic set packing problem.

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

`Stamps.jl`

from the book web-site, is given below:

```
#*****************************************************************************
# Stamp, "Mathematical Programming Modelling" (42112)
# Intro definitions
using JuMP
using HiGHS
#*****************************************************************************

#*****************************************************************************
# Data
include("stamp_bid_data.jl")
(B,S)=size(BidSets)
println("Bids: $B   stamps: $S")
#*****************************************************************************

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

# accept bid or not, yes if x[b]=1
@variable(stamps, x[1:B], Bin)

# Max sales profit
@objective(stamps, Max, sum( BidPrice[b] * x[b] for b=1:B) )

@constraint(stamps, [s=1:S],
            sum( BidSets[b,s]*x[b] for b=1:B) <= 1 )

print(stamps)
#*****************************************************************************

#*****************************************************************************
# solve
optimize!(stamps)
```

```
println("Termination status: $(termination_status(stamps))")
#*****************************************************************************

#*****************************************************************************
# Report results
let
if termination_status(stamps) == MOI.OPTIMAL
    println("RESULTS:")
    println("Stamps result: $(objective_value(stamps))")
    not_sold::Int8=0
    for s=1:S
        sold=sum(BidSets[b,s]*value(x[b]) for b=1:B)
        println("Stamp: $(s) : $(sold)")
        if sold < 1
            not_sold+=1
        end
    end
    println("Not sold stamps : $(not_sold)")
else
  println("  No solution")
end
end
#*****************************************************************************


#*****************************************************************************
println("Successfull end of $(PROGRAM_FILE)")
#*****************************************************************************
```

Comments to the Julia/JuMP program:

- Notice: We use the include statement to include the data from the book web-site

To require all stamps to be sold is simple to model, simple change $\leq$ to $=$ in the constraint. This however leads to an infeasible problem. Furthermore, since this is a restricted model, hence even **if** the new model was feasible, i.e. a solution existed, it would most likely get a lower sales profit, at best the same.