Food Festival

The Food Festival hiring problem is a variant of the classic OR problem: Graph Coloring. The shifts are the nodes in a graph and edges exists between nodes where the shifts conflicts. How few colours are needed to colour all the nodes ? Here the colurs are the workers to hire.

Problem

- Minimize the number of workers hired.

Sets

- $s \in Shifts = \{1, 2, 3, ..., 23, 24, 25\}$

Parameters

- $Conflicts_{s1,s2}$: A conflict incidence matrix, $Conflicts_{s1,s2} = 1$ means that two shifts cannot be done by the same shift worker.

Decision variables

- Worker w takes shift s: $x_{w,s}$.
- Hire worker w: y_w.

Model

Objective:

- Minimize the number of hired workers:

$$\sum_{w} y_w$$

Constraints:

- Ensure that all shifts are covered:

$$\sum_{w} x_{w,s} = 1 \ \forall \ s$$

- If a worker covers a shift, the worker has to be hired:

$$x_{w,s} \leq y_w \ \forall \ w, s$$

- Ensure that a worker cannot work shifts which are in conflict:

$$x_{w,s1} + x_{w,s2} \leq 1 \ \forall \ w, s1, s2 | s1 < s2 \wedge Conflict[s1, s2] == 1$$

Notice the conditions on the conflict ensuring constraint: Only the relevant constraints are generated.

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

`FoodFestival.jl`

from the book web-site, is given below:

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

#*****************************************************************
# PARAMETERS
include("FoodFestival_data.jl")
#*****************************************************************


#*****************************************************************
# Model
FF = Model(HiGHS.Optimizer)
W=S # maximal number of shift workers

@variable(FF, x[1:W,1:S],Bin) # if shift s is done by worker w
@variable(FF, y[1:W],Bin)     # worker w is hired
```

```
# Minimize number of workers
@objective(FF, Min,
           sum( y[w] for w=1:W)
           )

# Ensure that all shifts are worked
@constraint(FF, [s=1:S],
            sum( x[w,s] for w=1:W) == 1
            )

# Hire workers if used
@constraint(FF, [w=1:W,s=1:S],
            x[w,s] <= y[w]
            )

# Limit the same shifts
@constraint(FF, [w=1:W,s1=1:S,s2=1:S; s1<s2 && Conflict[s1,s2]==1],
            x[w,s1] + x[w,s2] <= 1
            )

#**************************************************************************

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

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