Project Scheduling

The project consists of a number of tasks which needs to be scheduled. However, some tasks can only be initiated when others have been completed. We use an **incidence** matrix, i.e. a matrix with only zero and one values, one for a precedence of one task over another.

Problem

- Minimize the completion time.

Sets

- $t \in Tasks = \{"Remove\ Stuff", "Shet\ Demolition", "Pipe\ Work", "Electric\ Work", "Flag\ Stones", "Shet\ Foundations", "Wood\ Frames", "Painting", "Roofing", "Roofing\ Felt", "Interior\ Installations", "Insert\ Stuff", "Hand\ Over"\}$

Parameters

- $Predecessors_{t2,t1}$: $Predecessors_{t2,t1} = 1$ if task $t2$ should be preceded by task $t1$
- $Duration_t$: The number of days it take to do task t

Decision variables

- The starting day of task t: $x_t \geq 0$.

Model

Objective:

- Minimize the starting time of the last task T ("$Hand\ Over$"):

$$x_T$$

Constraints:

- Precdence constraint:

$$x_{t1} + Duration_{t1} \leq x_{t2} \ \forall t1, t2 | Predecessors_{t2,t1} = 1$$

Notice the condition in the constraint: Only for tasks for which there is a precedense relation are the constraint defined.

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

`ProjectScheduling1.jl`

from the book web-site, is given below:

```
#*************************************************************************
# Project Scheduling Assignment, "Mathematical Programming Modelling" (42112)
using JuMP
using HiGHS
#*************************************************************************

#*************************************************************************
# PARAMETERS

# Tasks to be performed for the shet building
Tasks = ["Remove Stuff","Shet Demolition","Pipe Work","Electric Work", "Flag Stones","Shet P
T=length(Tasks)

# duration time in days to do the job
Duration = [1,3,5,3,7,4,3,1,1,1,3,1,0]

Predecessors=zeros(T,T)

#1

#2
Predecessors[2,1]=1

#3
Predecessors[3,2]=1

#4
Predecessors[4,2]=1

#5
```

```
Predecessors[5,3]=1
Predecessors[5,4]=1

#6
Predecessors[6,3]=1
Predecessors[6,4]=1

#7
Predecessors[7,6]=1

#8
Predecessors[8,7]=1

#9
Predecessors[9,7]=1

#10
Predecessors[10,9]=1

#11
Predecessors[11,9]=1

#12
Predecessors[12,9]=1

#13
Predecessors[13,5]=1
Predecessors[13,8]=1
Predecessors[13,10]=1
Predecessors[13,11]=1
Predecessors[13,12]=1
#*****************************************************************************

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

@variable(schedule, 0 <= x[t=1:T]) # start time of task t

# Minimize storage cost
@objective(schedule, Min, x[T])

# force predecessor relation
@constraint(schedule, [t1=1:T,t2=1:T; Predecessors[t2,t1]==1],
            x[t1] + Duration[t1] <= x[t2]
            )
```

```
print(schedule)
#*****************************************************************************

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

#*****************************************************************************
# Report results
println("-------------------------------------");
if termination_status(schedule) == MOI.OPTIMAL
    println("RESULTS:")
    println("Objective: $(objective_value(schedule))")
    for t=1:T
        println("Task: \t", Tasks[t], " starting-day: \t", value(x[t]), " Duration: \t", Du
    end
else
  println("  No solution")
end
println("-------------------------------------");
#*****************************************************************************
```

Comments to the Julia/JuMP program:

- Notice: The use of conditional constraint construction in the constraint.

In the second part of the assignment, the speedup of by paying extra is investigated. Notice, there is a budget of how much can be spend on the speedup. Here we only give the additions and changes to the model. The objective stays the same.

Problem

- Minimize the completion time, given a budget for speedup.
- $MaxSpeedUp_t$: The maximal number of days that task t can be speeded up.
- $OverTimePay_t$: The extra cost for each day which is speeded up.
- $Budget$: How much money we are willing to pay for faster completion. This can either be 5000, 10000 or ∞

Decision variables

- The start t: $0 \leq y_t \leq MaxSpeedUp_t$. Notice here we set the upper bound in the variable definition. This could also have been done using an additional constraint.

Constraints:

- Precedence constraint:

$$x_{t1} + Duration_{t1} - y_{t1} \leq x_{t2} \ \forall t1, t2 | Predecessors_{t2,t1} = 1$$

- Budget constraints for the overtime pay:

$$\sum_t OverTimePay_t \cdot y_{t1} \leq Budget \ \forall t$$

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

`ProjectScheduling2.jl`

from the book web-site, is given below:

```
#*************************************************************************
# Project Scheduling Assignment, "Mathematical Programming Modelling" (42112)
using JuMP
using Clp
#*************************************************************************

#*************************************************************************
# PARAMETERS

# Tasks to be performed for the shet building
Tasks = ["Remove Stuff","Shet Demolition","Pipe Work","Electric Work", "Flag Stones","Shet P
T=length(Tasks)

# duration time in days to do the job
Duration = [1,3,5,3,7,4, 3,1,1,1,3,1,0]

MaxSpeedUp=[0,1,3,1,4,2,2,0,0,0,2,0,0]

Budget=10000

OverTimePay=[0,500,2000,4000,1000,1000,1500,0,0,0,1500,0,0]

Predecessors=zeros(T,T)
```

```
#1

#2
Predecessors[2,1]=1

#3
Predecessors[3,2]=1

#4
Predecessors[4,2]=1

#5
Predecessors[5,3]=1
Predecessors[5,4]=1

#6
Predecessors[6,3]=1
Predecessors[6,4]=1

#7
Predecessors[7,6]=1

#8
Predecessors[8,7]=1

#9
Predecessors[9,7]=1

#10
Predecessors[10,9]=1

#11
Predecessors[11,9]=1

#12
Predecessors[12,9]=1

#13
Predecessors[13,5]=1
Predecessors[13,8]=1
Predecessors[13,10]=1
Predecessors[13,11]=1
Predecessors[13,12]=1

Budget=5000
#Budget=10000
```

```
#Budget=5000000
#*************************************************************************

#*************************************************************************
# Model
schedule = Model(Clp.Optimizer)

@variable(schedule, 0 <= x[t=1:T]) # start time of task t

@variable(schedule, 0 <= y[t=1:T] <= MaxSpeedUp[t] ) # save time

# Minimize storage cost
@objective(schedule, Min, x[T])

# force predecessor relation
@constraint(schedule, [t1=1:T,t2=1:T; Predecessors[t2,t1]==1],
            x[t1] + Duration[t1] - y[t1] <= x[t2]
            )

@constraint(schedule,
            sum( OverTimePay[t]*y[t] for t=1:T) <= Budget
            )

print(schedule)
#*************************************************************************


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

#*************************************************************************
# Report results
println("------------------------------------");
if termination_status(schedule) == MOI.OPTIMAL
    println("RESULTS:")
    println("Objective: $(objective_value(schedule))")
    println("sol x: ", value.(x))
    println("sol y: ", value.(y))
    for t=1:T
        println("Task: \t", Tasks[t], " starting-day: \t", value(x[t]), " Duration: \t", Du
    end
else
  println("  No solution")
```

```
end
println("---------------------------------------");
#*****************************************************************************
```

Comments to the Julia/JuMP program:

- Notice: You have to run the model for each different value of the *Budget* value.