Exercise 3

March 28, 2022

Bruno Kiyoshi Ynumaru

1 Process scheduling

1.1 Problem definition

A set of n jobs must be processed in a machine that can handle one job at a time. Task j needs pj hours to be processed. A directed and acyclic graph G = (V, E), with V = {1, . . . , n}, establishes a partial order for job processing in the machine. That is, if there exists a path i,j from i to j in G, then job i must be processed before job j. Given nonnegative weights wj , j = 1, . . . , n, in which order should we process the jobs in order to minimize the weighted sum of the start processing time of all jobs, while respecting the precedence order? For the modeling task that follows, sj is the instant that job j starts to be processed. Tasks: (a) Formulate the problem in mixed-integer linear programming using discrete and continuous variables. (b) Model the problem in AMPL and solve the instance given below, in which V = {1, . . . , 12}. Present the results.

job (j)	length (pj)	weight (wj)	Arcs (j,i)
1	3	5	(1,3)
2	2	3	
3	6	7	(3,12), (3,7)
4	2	6	
5	5	1	
6	4	2	(6,7)
7	4	8	
8	3	4	(8,6)
9	10	7	
10	1	1	(10,12)
11	8	6	
12	7	2	

1.1.1 Model

$\min \sum_{j=1}^{N} s_j \times w_j$
s.t. : $N \in \mathbb{Z}^+$
$s_j \geq (s_i + p_i).b_{i,j}$
$s_i \geq (s_j + p_j).\hat{b}_{i,j}$
$s_j \in \mathbb{R}_{\geq 0} \forall j$
$p_j \in \mathbb{R}_+$
$w_j \in \mathbb{R}_+$
$b_{i,j} \in \{0, 1\} \forall i, j$
$\hat{b}_{i,j} \in \{0, 1\} \forall i, j$
$b_{i,j} = 1 - \hat{b}_{i,j} \forall i, j$
$b_{j,j} = 0 \forall j$
$arc_{i,j} \in \{0, 1\} \forall i, j$
$b_{i,j} \geq arc_{i,j}$

1.1.2 Modeling (instance)

[1]:

```python
import gurobipy as gp
from gurobipy import GRB, Model
import pandas as pd
import numpy as np
```

Create supporting (instance) data

[2]:

```python
N = 12; # 'size' of the problem, here it's both the number of processes and␣
 ↪number of slots

# Create a list with the well indexes as given by the problem (prevents python's␣
 ↪0-indexing)
V = [item for item in range(1, N+1)]

# supporting data:
w = {j:[5,3,7,6,1,2,8,4,7,1,6,2][j-1] for j in V} # these are the given process␣
 ↪'weights' (importances)
p = {j:[3,2,6,2,5,4,4,3,10,1,8,7][j-1] for j in V} # these are the given process␣
 ↪time lenghts

"""
'dependance table'
     p1 p2 p3 p4
p1  [0  1  0  0] <- p1 must come before p2
p2  [0  0  0  0]
p3  [1  0  0  0] <- p3 must come before p1
p4  [0  0  0  0]
"""
D = np.zeros((N,N))
```

```python
arcs = ((1,3),(3,12),(3,7),(6,7),(8,6),(10,12)) # these arcs are in order (i,j), where i depends on j
for arc in arcs:
    x=int(arc[0]) - 1 # minus one is because of python 0 indexing
    y=int(arc[1]) - 1 # minus one is because of python 0 indexing
    D[x][y] = 1
print('dependance table\n',D)
```

```
dependance table
 [[0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Model creation

[3]:

```python
m = Model("process_scheduling")

# Decision variables:
# 1)represents the starting time of each process
'''
s: [1, 3, 4, ...] process 1 starts at time=1, process 2 starts at time=3...
'''
s = m.addVars(V, vtype=GRB.CONTINUOUS, name="s")

# 2)decision varibles. If B[i,j]=1, then process i comes before process j
"""
'comes before table'
     p1 p2 p3 p4
p1  [0  1  1  1] p1 comes before p2, p3 and p4 - first
p2  [0  0  0  0] p2 comes before no one-last
p3  [0  1  0  1] p3 comes before p2 and p4 - second
p4  [0  1  0  0] p4 - third
p1->p3->p4->p2
"""

B=m.addVars(V, V, vtype=GRB.BINARY, name='B')
NB=m.addVars(V, V, vtype=GRB.BINARY, name='NB') # negation of B

m.update()
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2022-05-20
```

Constraints

[4]:

```
#Constraints over the binary tables
for i in V:
    for j in V:
        m.addConstr(B[i,j]==1-NB[i,j], name=f"BNB{i}_{j}") # if job i comes␣
 ↪before job j, then job j cannot come before job i

m.update()
```

[5]:

```
# Constraints related to the
# 1)non-overlaping usages of resources
for j in V:
    for i in (set(V)-set([j])):
        m.addConstr(s[j]>=(s[i]+p[i])*B[i,j])
        m.addConstr(s[i]>=(s[j]+p[j])*NB[i,j])

# 2)given process precedence requirements
for i in V:
    for j in V:
        Di = i - 1
        Dj = j - 1
        if D[Di][Dj] == 1:
            m.addConstr(B[i,j]==1,name=f'arc({i},{j})')

# 3) a job cannot happen before itself
for i in V:
    m.addConstr(B[i,i]==0)

m.update()
```

Solution

[6]:

```
m.setObjective(sum([s[j]*w[j] for j in V]), GRB.MINIMIZE)

m.update()
```

[7]:

```
m.optimize()
```

```
Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (win64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 162 rows, 300 columns and 306 nonzeros
Model fingerprint: 0x7ca0adbe
Model has 264 quadratic constraints
Variable types: 12 continuous, 288 integer (288 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  QMatrix range    [1e+00, 1e+00]
  QLMatrix range   [1e+00, 1e+01]
  Objective range  [1e+00, 8e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 156 rows and 162 columns
Presolve time: 0.00s
Presolved: 762 rows, 894 columns, 2154 nonzeros
Presolved model has 504 SOS constraint(s)
Variable types: 516 continuous, 378 integer (378 binary)

Root relaxation: objective 2.478783e+02, 506 iterations, 0.01 seconds (0.00 work
units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  247.87830    0  244          -  247.87830      -     -    0s
     0     0  274.41538    0  228          -  274.41538      -     -    0s
     0     0  274.41538    0  228          -  274.41538      -     -    0s
     0     0  282.19077    0  224          -  282.19077      -     -    0s
     0     0  282.19077    0  193          -  282.19077      -     -    0s
     0     0  284.40157    0  161          -  284.40157      -     -    0s
     0     0  284.40157    0  164          -  284.40157      -     -    0s
     0     0  285.50287    0  197          -  285.50287      -     -    0s
     0     0  286.81818    0  198          -  286.81818      -     -    0s
     0     0  286.84706    0  211          -  286.84706      -     -    0s
     0     0  289.66241    0  210          -  289.66241      -     -    0s
     0     0  290.34697    0  206          -  290.34697      -     -    0s
     0     0  290.60709    0  232          -  290.60709      -     -    0s
     0     0  290.86322    0  228          -  290.86322      -     -    0s
     0     0  290.86322    0  193          -  290.86322      -     -    0s
     0     2  290.86322    0  190          -  290.86322      -     -    0s
*  257   113              70    1129.0000000  374.20000  66.9%   4.7    0s
H  293   146                    1120.0000000  374.20000  66.6%   4.6    0s
H  324   145                     902.0000000  374.20000  58.5%   5.0    0s
*  329   145              78     898.0000000  374.20000  58.3%   5.0    0s
H  353   158                     869.0000000  374.20000  56.9%   5.1    0s
H 1204   509                     868.0000000  423.00000  51.3%   5.1    0s
H 1652   625                     865.0000000  427.00000  50.6%   8.2    1s
* 2373   666              54     861.0000000  466.20000  45.9%   9.8    2s
H 2398   633                     860.0000000  466.20000  45.8%   9.8    2s
H 3600   840                     859.0000000  509.03597  40.7%   9.8    3s
H 4532  1115                     856.0000000  529.66447  38.1%   9.9    3s
  6164  1489  829.00000   42    7  856.00000  555.20955  35.1%  10.0    5s
 17490  2545  684.00000   41   13  856.00000  656.00000  23.4%   9.7   10s
 30835  1776 infeasible   50       856.00000  748.00000  12.6%   9.4   15s
```

```
Cutting planes:
  Gomory: 3
  MIR: 27

Explored 37097 nodes (340886 simplex iterations) in 16.78 seconds (7.12 work
units)
Thread count was 4 (of 4 available processors)

Solution count 10: 856 859 860 ... 1120

Optimal solution found (tolerance 1.00e-04)
Best objective 8.560000001483e+02, best bound 8.560000001483e+02, gap 0.0000%
```

[8]:

```
for j in V:
    start=round(s[j].x)
    end = start+p[j]
    print(f'Process {j} starts at {start} and ends at {end}')
```

```
Process 1 starts at 2 and ends at 5
Process 2 starts at 5 and ends at 7
Process 3 starts at 10 and ends at 16
Process 4 starts at 0 and ends at 2
Process 5 starts at 50 and ends at 55
Process 6 starts at 16 and ends at 20
Process 7 starts at 20 and ends at 24
Process 8 starts at 7 and ends at 10
Process 9 starts at 33 and ends at 43
Process 10 starts at 24 and ends at 25
Process 11 starts at 25 and ends at 33
Process 12 starts at 43 and ends at 50
```