Stefanno Da Silva and Julie Yu

February 1st 2018

A1_Problem1

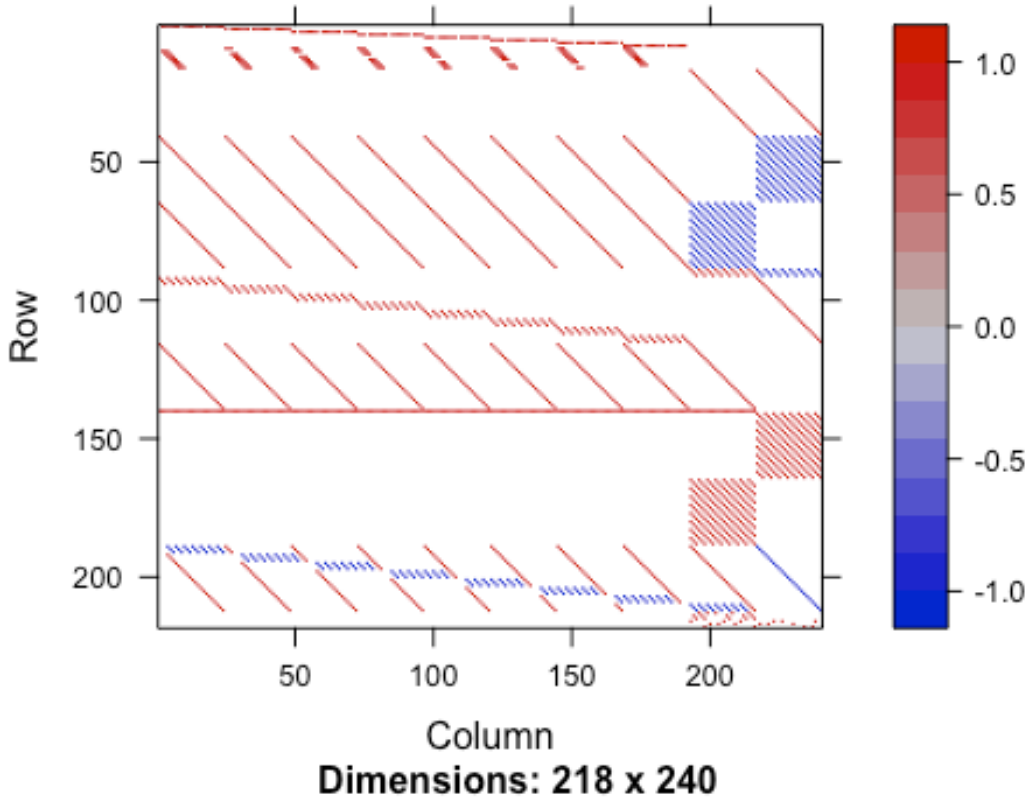
```
library("gurobi")

## Loading required package: slam

library("Matrix")
library("igraph")

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library("rdist")


n <- 8
k <- 3

M <- 100000
T.ij = matrix(c(M, M, 60, 90, M, 180, M, M, M, M, 30, 60, M, 150, M, M,
M, M, M, M, 30, M, 90, M, M, M, M, M, M, M, 30, M, M, M, M, M, M, 30, M
, M, M, M, M, M, M, M, 90, M, M, M, M, M, M, M, 60, M, M, M, M, M, M
, M, M), nrow = n, ncol=n, byrow = TRUE)
F.ik.S.ik = c(rep(0, n*k*2))
cvec = c(rep(0, n*n*k),F.ik.S.ik)
col = 1
for(i in 1:n){
  for(j in 1:n){
    for(K in 1:k){
      cvec[col] = T.ij[i,j]
      col = col + 1
    }
  }
}

bvec = c(rep(1, n), rep(1, n), rep(1, n*k), rep(0, n*k), rep(0, n*k), r
ep(0, k), rep(1, n*k), rep(1, n*k), 8, rep(1, n*k), rep(1, n*k), rep(0,
n*k), rep(0, 6) )
```

```
dir = c(rep("<=", n), rep("<=", n),  rep("<=", n*k), rep(">=", n*k), re
p(">=", n*k), rep("=", k), rep("<=", n*k), rep("<=", n*k), "=", rep("<=
", n*k), rep("<=", n*k), rep("=", n*k), rep("=", 6))

Amat = matrix(0, nrow=(n*2+(n*k)*8+k+7), ncol=(n*n*k + n*k*2))
#contraint 1
for (i in 1:n) {
  Amat[i, ((i - 1) * n*k + 1):(i * n*k)] = 1
  Amat[i, ((i - 1) * n*k + i)] = 0
  Amat[i, ((i - 1) * n*k + i + n)] = 0
  Amat[i, ((i - 1) * n*k + i + n*2)] = 0
}
#contraint 2
for (i in 1:n) {
  Amat[i+n, seq(i,by =n*k, length.out = n)] = 1
  Amat[i+n, seq(i+1,by =n*k, length.out = n)] = 1
  Amat[i+n, seq(i+2,by =n*k, length.out = n)] = 1
  Amat[i+n, ((i - 1) * n*k + 1):(i * n*k)] = 0
}
#contraint 3
row = 1
for (i in 1:n) {
  for(K in 1:k){
    Amat[row+n*2, seq(from=row+n*n*k,by =n*k, length.out = 2)] = 1
    row = row + 1
  }
}
#contraint 4
row = 1
col = 1
for (i in 1:n) {
  for(K in 1:k){
    Amat[row+n*2+n*k, seq(row,by =n*k, length.out = n)] = 1
    Amat[row+n*2+n*k, seq(from=n*n*k+n*k+col,by =k, length.out = n)] =
-1
    row = row + 1
    col = col + 1
  }
  col = 1
}
#contraint 5
row = 1
col = 1
for (i in 1:n) {
  for(K in 1:k){
    Amat[row+n*2+n*k*2, seq(row,by =n*k, length.out = n)] = 1
    Amat[row+n*2+n*k*2, seq(from=n*n*k+col,by =k, length.out = n)] = -1
    row = row + 1
    col = col + 1
  }
```

```
  col = 1
}
#contraint 6
for (i in 1:(k)) {
  Amat[i+n*2+n*k*3, seq(from=n*n*k+i,by =k, length.out = n)] = 1
  Amat[i+n*2+n*k*3, seq(from=n*n*k+n*k+i,by =k, length.out = n)] = -1
}

#contraint 7
Count = 1
Count2 = 1
for(i in 1:n){
  for(K in 1:k){
    Amat[Count+n*2+n*k*3+k,seq(from=Count2,by =k, length.out = n)] = 1
    Amat[Count+n*2+n*k*3+k,216+Count] = 1
    Count = Count + 1
    Count2 = Count2 + 1
  }
  Count2 = Count2 + (n-1)*k
}

#contraint 8
Count = 1
Count2 = 1
for(i in 1:n){
  for(K in 1:k){
    Amat[Count+n*2+n*k*4+k,seq(from=Count2,by =n*k, length.out = n)] =
1
    Amat[Count+n*2+n*k*4+k,192+Count] = 1
    Count = Count + 1
    Count2 = Count2 + 1
  }
  #Count2 = Count2 + (n-1)*k
}

#Constraint 9: make sure that when you sum all F and all X it equals to
8
Amat[n*2+n*k*5+k+1,(1):(n*n*k + n*k)] = 1

#Constraint 10: each crew can only have first flight once
row = 1
col = 1
for (i in 1:n) {
  for(K in 1:k){
    Amat[row+n*2+n*k*5+k+1, seq(from=n*n*k+n*k+col,by =k, length.out =
n)] = 1
    row = row + 1
    col = col + 1
  }
```

```
  col = 1
}
#Constraint 11: each crew can only have last flight once
row = 1
col = 1
for (i in 1:n) {
  for(K in 1:k){
    Amat[row+n*2+n*k*6+k+1, seq(from=n*n*k+col,by =k, length.out = n)]
= 1
    row = row + 1
    col = col + 1
  }
  col = 1
}
#Constraint 12: make sure that if a crew goes from i to j then j to som
ething else should still be done by the same crew (His paper)
row = 1
offSet = 0
offSet2 = 0
for (i in 1:n) {
  for(K in 1:k){
     Amat[row+n*2+n*k*7+k+1, seq(row,by =n*k, length.out = n)] = 1
     Amat[row+n*2+n*k*7+k+1, seq(K + offSet2,by =n*k, length.out = 1)]
= 0
     Amat[row+n*2+n*k*7+k+1, seq(row + offSet + k,by =k, length.out = n
-1)] = -1
     Amat[row+n*2+n*k*7+k+1, seq(from=row+n*n*k,by =n*k, length.out = 1
)] = 1
     Amat[row+n*2+n*k*7+k+1, seq(from=row+n*n*k+n*k,by =n*k, length.out
= 1)] = -1
    row = row + 1
  }
  offSet = offSet + 24
  offSet2 = offSet2 + 27
}

#constraint 13: make sure that if someone starts somewhere it also need
s to end in the same city
Crew1NotF = c(2,3,4,6,8)
for(i in 1:length(Crew1NotF)){
  Amat[n*2+n*k*8+k+2, n*n*k+(Crew1NotF[i]-1)*3+1] = 1
}
Crew2NotF = c(1,2,5,7,8)
for(i in 1:length(Crew2NotF)){
  Amat[n*2+n*k*8+k+3, n*n*k+(Crew2NotF[i]-1)*3+2] = 1
}
Crew3NotF = c(1,2,5,7,8)
for(i in 1:length(Crew3NotF)){
  Amat[n*2+n*k*8+k+4, n*n*k+(Crew3NotF[i]-1)*3+3] = 1
}
```

```
#Constraint 14: make sure that each crew can only start in one given ci
ty
Crew1F = c(1,5,7)
for(i in 1:length(Crew1F)){
  Amat[n*2+n*k*8+k+5, n*n*k+(Crew1F[i]-1)*3+1] = 1
}
Crew1L = c(3,4,8)
for(i in 1:length(Crew1L)){
  Amat[n*2+n*k*8+k+5, n*n*k+n*k+(Crew1L[i]-1)*3+1] = 1
}
Crew2F = c(3,4,6)
for(i in 1:length(Crew2F)){
  Amat[n*2+n*k*8+k+6, n*n*k+(Crew2F[i]-1)*3+2] = 1
}
Crew2L = c(1,2,5)
for(i in 1:length(Crew2L)){
  Amat[n*2+n*k*8+k+6, n*n*k+n*k+(Crew2L[i]-1)*3+2] = 1
}
Crew3F = c(2,8)
for(i in 1:length(Crew3F)){
  Amat[n*2+n*k*8+k+7, n*n*k+(Crew3F[i]-1)*3+3] = 1
}
Crew3L = c(6,7)
for(i in 1:length(Crew3L)){
  Amat[n*2+n*k*8+k+7, n*n*k+n*k+(Crew3L[i]-1)*3+3] = 1
}


image(Matrix(Amat))
```




```
myLP = list()
myLP$obj = cvec
myLP$A = Amat
myLP$sense = dir
myLP$rhs = bvec
myLP$vtypes = "B"
myLP$ub = 1

mysol = gurobi(myLP)

## Warning for adding variables: zero or small (< 1e-13) coefficients,
ignored
## Optimize a model with 218 rows, 240 columns and 2647 nonzeros
## Variable types: 0 continuous, 240 integer (240 binary)
## Coefficient statistics:
##   Matrix range     [1e+00, 1e+00]
##   Objective range  [3e+01, 1e+05]
##   Bounds range     [1e+00, 1e+00]
##   RHS range        [1e+00, 8e+00]
## Found heuristic solution: objective 800000.00000
## Presolve removed 137 rows and 72 columns
## Presolve time: 0.02s
## Presolved: 81 rows, 168 columns, 1180 nonzeros
```

```
## Variable types: 0 continuous, 168 integer (168 binary)
##
## Root relaxation: objective 2.502400e+05, 55 iterations, 0.00 seconds
##
##     Nodes    |    Current Node    |     Objective Bounds      |
Work
##  Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/N
ode Time
##
##      0     0 250240.000    0    6 800000.000 250240.000  68.7%     -
0s
## H    0     0                    400180.00000 250240.000  37.5%     -
0s
##      0     0 250240.000    0   12 400180.000 250240.000  37.5%     -
0s
## H    0     0                    400150.00000 250240.000  37.5%     -
0s
## *    0     0               0    300180.00000 300180.000  0.00%     -
0s
##
## Cutting planes:
##   Gomory: 2
##
## Explored 1 nodes (88 simplex iterations) in 0.04 seconds
## Thread count was 4 (of 4 available processors)
##
## Solution count 4: 300180 400150 400180 800000
##
## Optimal solution found (tolerance 1.00e-04)
## Best objective 3.001800000000e+05, best bound 3.001800000000e+05, ga
p 0.0000%

mysol$objval

## [1] 300180

mysol$x

##   [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 0 0
0 0 0 1
##  [36] 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
##  [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0
## [106] 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
## [141] 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
## [176] 1 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 0
## [211] 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
```