Optimization HW10 Ang Zhou

Question 1

(a)

To formulate the problem into a min-cost network flow problem, I decided to assign costs on each combination of referees and paper. Use ‘i’ to denote 71 papers, and ‘j’ to denote 21 papers.

If the referees are assigned to papers that they said “yes, I can definitely review this paper”, the costs are 1. If the referees are assigned to papers that they said “I can maybe review this paper”, the costs are 10. If the referees are assigned to papers that they said “No, I do not want to review this paper” the costs are 100. If the referees are assigned to papers that they said “I have a conflict of interest with this paper, so it is unethical for me to review this paper” the costs are 10^6, since we definitely do not want a referee to be assigned to a paper if he/she said that.

The cost table would be look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Yes | Maybe | No | Definitely Not |
| Costs | 1 | 10 | 100 | 10^6 |

Let Cij denote the cost that assign paper ‘i’ to referee ‘j’.

Let Xij denote the true/false that paper ‘i’ is assigned to referee ‘j’, Xij=1 means paper ‘i’ is assigned to referee ‘j’.

Our objective function would be:

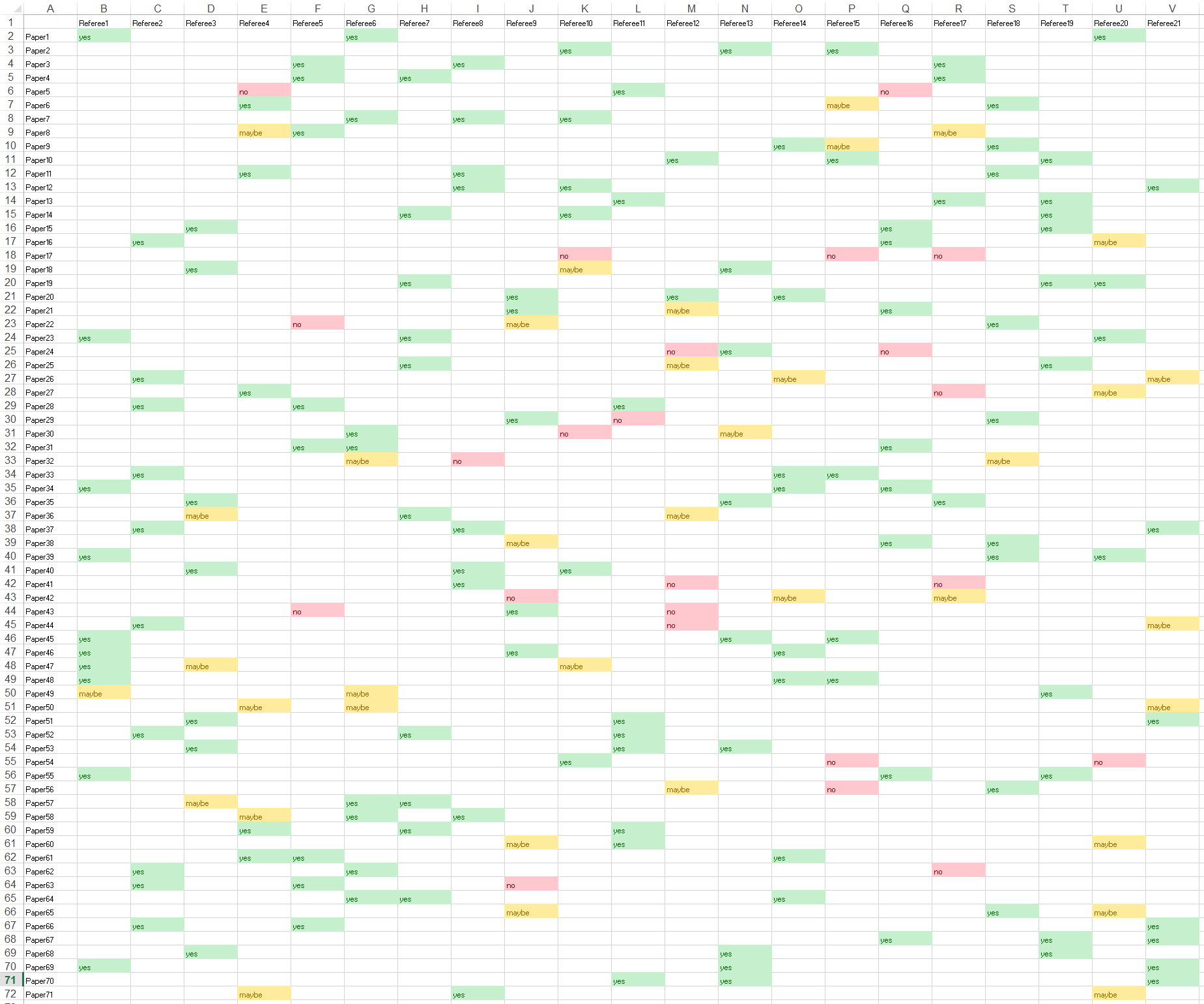
Min

Since each paper need to have 3 referees and each referee need to review about 10~11 papers

Our constraints are represented as following:

1. , where i belongs to [1,71]
2. , where j belongs to [1,21]
3. , where j belongs to [1,21]
4. Xij 0
5. Xij 1

(b)After using the gurobi to solve this optimization problem, we got the following result:



Code:

from gurobipy import \*

import xlwt

import xlrd

# loading data

f = xlrd.open\_workbook('paper\_data.xlsx')

sheet = f.sheet\_by\_index(0)

# create a new model

myModel = Model("HW10\_Q1")

# create decision vats and integrate them into the model

nrows = sheet.nrows-1

ncols = sheet.ncols-1

cost = [[0 for i in range(ncols)] for j in range(nrows)]

myVars = [[0 for i in range(ncols)] for j in range(nrows)]

for i in range(nrows):

for j in range(ncols):

curVar = myModel.addVar(vtype=GRB.INTEGER, name='X' + str(i+1) + ',' + str(j+1))

myVars[i][j] = curVar

arc = sheet.cell(i+1,j+1).value

if arc == "yes":

cost[i][j] = 1

if arc == "maybe":

cost[i][j] = 10

if arc == "no":

cost[i][j] = 100

if arc == "conflict":

cost[i][j] = 1000000

myModel.update()

# create a linear expression for the objective

objExpr = LinExpr()

for i in range(nrows):

for j in range(ncols):

curVar = myVars[i][j]

objExpr += cost[i][j]\*curVar

myModel.setObjective(objExpr, GRB.MINIMIZE)

myModel.update()

# Constraint for each paper having exactly three referees to review

for i in range(nrows):

constExpr = LinExpr()

for j in range(ncols):

curVar = myVars[i][j]

constExpr += curVar

myModel.addConstr(lhs=constExpr, sense=GRB.EQUAL, rhs=3, name="paper" + str(i+1))

# Constraint for each referee having at least 10 paper to review

for j in range(ncols):

constExpr = LinExpr()

for i in range(nrows):

curVar = myVars[i][j]

constExpr += curVar

myModel.addConstr(lhs=constExpr, sense=GRB.GREATER\_EQUAL, rhs=10, name="referee\_lowerbound" + str(j+1))

# Constraint for each referee having at most 11 paper to review

for j in range(ncols):

constExpr = LinExpr()

for i in range(nrows):

curVar = myVars[i][j]

constExpr += curVar

myModel.addConstr(lhs=constExpr, sense=GRB.LESS\_EQUAL, rhs=11, name="referee\_upperbound" + str(j+1))

# boundary

for j in range(ncols):

for i in range(nrows):

constExpr = LinExpr()

constExpr = myVars[i][j]

myModel.addConstr(lhs=constExpr, sense=GRB.LESS\_EQUAL, rhs=1, name="boundary" + str(i+1) + ',' + str(j+1))

# integrate objective and constraints into the model

myModel.update()

# write the model in a file to make sure it is constructed correctly

myModel.write(filename="HW10\_Q1.lp")

# optimize the model

myModel.optimize()

# save result to xls file

allVars = myModel.getVars()

writebook = xlwt.Workbook()

sheet1 = writebook.add\_sheet('test')

for i in range(71):

for j in range(21):

if(allVars[21\*i+j].x == 1):

sheet1.write(i+1,j+1,sheet.cell(i+1,j+1).value)

writebook.save('result.xls')