Week 4 Homework

1. Problem 1 (16 pts) This problem involves building a transportation model. Instead of having suppliers ship directly to customers, this problem has factories that ship to warehouses and warehouses that ship to stores. (Think Walmart distribution centers). There are different per unit costs associated with shipping between factories and warehouses and also between warehouses and stores. Your job is to minimize the total shipping costs subject to the following constraints
   1. The total supply of each product is the same as the total demand
   2. The total over all products that may be shipped along a route is constrained by CapacityFW (from Factories to Warehouses) and Capacity WS (from Warehouses to Stores)
   3. The total number of all products that may be stored in a Warehouse is limited by MaxStorage
   4. For each product and for each warehouse the total product shipped from Factories to that Warehouse is the same as the total product shipped from that Warehouse to stores.

Here is the R script (this is the file transpStoreBigWrite.R in the solutions folder)

require('XLConnect') # install if necessary

# also install XLConnectJars if not installed as dependency

ws <- 'transpStoreBigSheet.xlsx'

# read dataframes from Excel spreadsheet

# you'll have to change the sheets and regions

dfFactories <- readWorksheetFromFile(ws,sheet=1,region='A1:A6')

dfWarehouses <- readWorksheetFromFile(ws,sheet=1,region='B1:B11')

dfStores <- readWorksheetFromFile(ws,sheet=1,region='C1:C21')

dfProducts <- readWorksheetFromFile(ws,sheet=1,region='D1:D6')

dfTableRoutesFW <- readWorksheetFromFile(ws,sheet=1,region='G2:J71')

dfTableRoutesWS <- readWorksheetFromFile(ws,sheet=1,region='L2:O202')

dfSupply <- readWorksheetFromFile(ws,sheet=1,region='Q2:S15')

dfDemand <- readWorksheetFromFile(ws,sheet=1,region='V2:X102')

dfCapacityFW <- readWorksheetFromFile(ws,sheet=1,region='E1:E2')

dfCapacityWS <- readWorksheetFromFile(ws,sheet=1,region='E4:E5')

dfMaxStorage <- readWorksheetFromFile(ws,sheet=1,region='E7:E8')

# change to your directory or set working directory in R

#setwd('~//Users/jbaggett/Google Drive/Gdrive\_snap\_May\_15/MS Data Science/DS 775/Class Materials/Weekly Content/Week\_04')

filename <- 'transpStoreBig.dat'

cat('//--------------------------------------------------\n',file=filename)

con <- file(filename) # get connection to file to close later

cat('// transportation file produced by reading \n',file=filename,append=T)

cat('// excel file into R using XLConnect \n',file=filename,append=T)

cat('// and using R commands to output textfile \n',file=filename,append=T)

cat('//--------------------------------------------------\n\n',file=filename,append=T)

# some helper functions for writing in the right formats for .dat file

writeSet <- function( varName, df, filename ){

cat( varName, file = filename, append = T )

cat( ' = { ', file = filename, append = T )

numRows <- dim(df)[1]

for (j in 1:numRows){

cat( df[j,1], file = filename, append = T )

cat( ' ', file = filename, append = T )

}

cat('};\n', file = filename, append = T )

}

writeNumber <- function( varName, x, filename ){

cat( varName, file = filename, append = T)

cat( ' = ', file = filename, append = T)

cat( x, file = filename, append = T)

cat( ';\n', file = filename, append = T)

}

writeTuplesTable <- function( varName, df, filename){

cat( varName, file = filename, append = T)

cat( ' = {\n', file = filename, append = T)

numRows <- dim( df )[1]

for (j in 1:numRows){

cat(' < ', file = filename, append = T)

write.table( df[j,], file = filename, append = T, quote = F, sep = ', ',

row.names = F, col.names = F, eol = ' >\n' )

}

cat('};\n\n', file = filename, append = T)

}

writeTuplesArray <- function( varName, df, filename){

cat( varName, file = filename, append = T)

cat( ' = #[\n', file = filename, append = T)

numRows <- dim( df )[1]

numCols <- dim( df )[2]

tupleLen <- numCols - 1

for (j in 1:numRows){

cat(' < ', file = filename, append = T)

write.table( df[j,1:tupleLen], file = filename, append = T, quote = F, sep = ', ',

row.names = F, col.names = F, eol = ' >: ' )

cat( df[j,numCols] , file = filename, append = T )

cat( '\n', file = filename, append = T )

}

cat(']#;\n\n', file = filename, append = T)

}

# write the file and close it

writeSet( 'Factories', dfFactories, filename)

writeSet( 'Warehouses', dfWarehouses, filename)

writeSet( 'Stores', dfStores, filename)

writeSet( 'Products', dfProducts, filename)

writeNumber( 'CapacityFW', dfCapacityFW[1,1],filename)

writeNumber( 'CapacityWS', dfCapacityWS[1,1],filename)

writeNumber( 'MaxStorage', dfMaxStorage[1,1],filename)

writeTuplesTable( 'TableRoutesFW', dfTableRoutesFW, filename)

writeTuplesTable( 'TableRoutesWS', dfTableRoutesWS, filename)

writeTuplesArray( 'Supply', dfSupply, filename)

writeTuplesArray( 'Demand', dfDemand, filename)

close(con) # close connection to file

Here is the model file (transpStoreBig.mod in the solutions folder):

{string} Factories = ...;

{string} Warehouses = ...;

{string} Stores = ...;

{string} Products = ...;

float CapacityFW = ...;

float CapacityWS = ...;

float MaxStorage = ...;

tuple tableRoutesType {

string p;

string o;

string d;

float cost;

}

{tableRoutesType} TableRoutesFW = ...;

{tableRoutesType} TableRoutesWS = ...;

tuple connection {

string o;

string d;

}

tuple route {

string p;

connection e;

}

{route} RoutesFW = { < p,<o,d> > | <p,o,d,c> in TableRoutesFW };

{route} RoutesWS = { < p,<o,d> > | <p,o,d,c> in TableRoutesWS };

{connection} ConnectionsFW = { c | <p,c> in RoutesFW };

{connection} ConnectionsWS = { c | <p,c> in RoutesWS };

tuple supply{

string p;

string o;

}

{supply} Suppliers = { <p,c.o> | <p,c> in RoutesFW };

float Supply[Suppliers] = ...;

tuple customer {

string p;

string d;

}

{customer} Customers = { <p,c.d> | <p,c> in RoutesWS };

float Demand[Customers] = ...;

float CostFW[RoutesFW] = [ <t.p,<t.o,t.d>>:t.cost | t in TableRoutesFW ];

float CostWS[RoutesWS] = [ <t.p,<t.o,t.d>>:t.cost | t in TableRoutesWS ];

{string} Orig[p in Products] = { c.o | <p,c> in RoutesFW };

{string} Ware[p in Products] = { c.d | <p,c> in RoutesFW };

{string} Dest[p in Products] = { c.d | <p,c> in RoutesWS };

{connection} CPsFW[p in Products] = { c | <p,c> in RoutesFW };

{connection} CPsWS[p in Products] = { c | <p,c> in RoutesWS };

assert forall( p in Products )

sum( o in Orig[p] )

Supply[<p,o>] == sum( d in Dest[p] ) Demand[<p,d>];

dvar float+ TransFW[RoutesFW];

dvar float+ TransWS[RoutesWS];

dexpr float StoreW[w in Warehouses] =

sum(p in Products) sum(<o,w> in CPsFW[p]) TransFW[<p,<o,w>>];

constraint ctSupply[Products][Factories];

constraint ctDemand[Products][Stores];

constraint ctConnect[Products][Warehouses];

minimize

sum( l in RoutesFW ) CostFW[l] \* TransFW[l]

+ sum( k in RoutesWS) CostWS[k] \* TransWS[k];

subject to {

forall( p in Products , o in Orig[p] )

ctSupply[p][o]:

sum( <o,d> in CPsFW[p] )

TransFW[< p,<o,d> >] == Supply[<p,o>];

forall( p in Products , d in Dest[p] )

ctDemand[p][d]:

sum( <o,d> in CPsWS[p] )

TransWS[< p,<o,d> >] == Demand[<p,d>];

forall( p in Products, w in Ware[p])

ctConnect[p][w]:

sum( <o,w> in CPsFW[p]) TransFW[<p,<o,w>>]

- sum( <w,d> in CPsWS[p]) TransWS[<p,<w,d>>] == 0;

forall( c in ConnectionsFW )

ctCapacityFW:

sum( <p,c> in RoutesFW )

TransFW[<p,c>] <= CapacityFW;

forall( c in ConnectionsWS )

ctCapacityWC:

sum( <p,c> in RoutesWS )

TransWS[<p,c>] <= CapacityWS;

forall( w in Warehouses )

ctStorage:

sum( p in Products, <o,w> in CPsFW[p] ) TransFW[<p,<o,w>>] <= MaxStorage;

}

execute DISPLAY {

writeln("Transport Amounts FW:");

for (var r in RoutesFW) {

writeln(" ", r.p , ":" , r.e.o , "->" , r.e.d , " " , TransFW[r] );

}

writeln("\nStorage in Warehouse:");

for (var w in Warehouses){

writeln( w," : ",StoreW[w] )

}

writeln("Transport Amounts WS:")

for (var r in RoutesWS) {

writeln(" ", r.p , ":" , r.e.o , "->" , r.e.d , " " , TransWS[r] );

}

}

Here is the solution output from the scripting window:

// solution (optimal) with objective 45360

Transport Amounts FW:

pA:fA->wA 240

pA:fA->wB 280

pA:fA->wC 70

pA:fA->wD 110

pA:fB->wC 0

pA:fB->wD 120

pA:fB->wE 260

pA:fB->wF 180

pA:fC->wE 110

pA:fC->wF 0

pA:fC->wG 170

pA:fC->wH 180

pA:fC->wI 120

pA:fC->wJ 220

pB:fA->wA 260

pB:fA->wB 240

pB:fA->wC 0

pB:fA->wD 0

pB:fB->wC 110

pB:fB->wD 260

pB:fB->wE 540

pB:fB->wF 90

pB:fC->wE 0

pB:fC->wF 0

pB:fC->wG 60

pB:fC->wH 160

pB:fC->wI 140

pB:fC->wJ 240

pC:fA->wA 10

pC:fA->wB 130

pC:fA->wC 40

pC:fA->wD 220

pC:fA->wE 0

pC:fA->wF 0

pC:fB->wE 170

pC:fB->wF 10

pC:fB->wG 80

pC:fB->wH 80

pC:fB->wI 70

pC:fB->wJ 260

pD:fD->wA 0

pD:fD->wB 60

pD:fD->wC 30

pD:fD->wD 70

pD:fD->wE 140

pD:fD->wF 0

pD:fD->wG 0

pD:fE->wE 0

pD:fE->wF 0

pD:fE->wG 50

pD:fE->wH 50

pD:fE->wI 40

pD:fE->wJ 140

pE:fC->wA 420

pE:fC->wB 450

pE:fC->wC 530

pE:fD->wA 0

pE:fD->wB 0

pE:fD->wC 0

pE:fD->wD 210

pE:fD->wE 790

pE:fD->wF 0

pE:fD->wG 0

pE:fE->wE 290

pE:fE->wF 0

pE:fE->wG 340

pE:fE->wH 380

pE:fE->wI 230

pE:fE->wJ 520

Storage in Warehouse:

wA : 930

wB : 1160

wC : 780

wD : 990

wE : 2300

wF : 280

wG : 700

wH : 850

wI : 600

wJ : 1380

Transport Amounts WS:

pA:wA->sA 150

pA:wA->sB 90

pA:wA->sC 0

pA:wA->sD 0

pA:wB->sC 80

pA:wB->sD 140

pA:wB->sE 60

pA:wB->sF 0

pA:wC->sE 0

pA:wC->sF 70

pA:wC->sG 0

pA:wC->sH 0

pA:wD->sG 110

pA:wD->sH 120

pA:wD->sI 0

pA:wD->sJ 0

pA:wE->sI 90

pA:wE->sJ 200

pA:wE->sK 0

pA:wE->sL 80

pA:wF->sK 180

pA:wF->sL 0

pA:wF->sM 0

pA:wF->sN 0

pA:wG->sM 60

pA:wG->sN 110

pA:wG->sO 0

pA:wG->sP 0

pA:wH->sO 120

pA:wH->sP 60

pA:wH->sQ 0

pA:wH->sR 0

pA:wI->sQ 70

pA:wI->sR 50

pA:wI->sS 0

pA:wI->sT 0

pA:wJ->sS 100

pA:wJ->sT 120

pA:wJ->sA 0

pA:wJ->sB 0

pB:wA->sA 160

pB:wA->sB 100

pB:wA->sC 0

pB:wA->sD 0

pB:wB->sC 70

pB:wB->sD 130

pB:wB->sE 40

pB:wB->sF 0

pB:wC->sE 30

pB:wC->sF 80

pB:wC->sG 0

pB:wC->sH 0

pB:wD->sG 100

pB:wD->sH 110

pB:wD->sI 0

pB:wD->sJ 50

pB:wE->sI 100

pB:wE->sJ 160

pB:wE->sK 190

pB:wE->sL 90

pB:wF->sK 0

pB:wF->sL 0

pB:wF->sM 50

pB:wF->sN 40

pB:wG->sM 0

pB:wG->sN 60

pB:wG->sO 0

pB:wG->sP 0

pB:wH->sO 110

pB:wH->sP 50

pB:wH->sQ 0

pB:wH->sR 0

pB:wI->sQ 60

pB:wI->sR 80

pB:wI->sS 0

pB:wI->sT 0

pB:wJ->sS 110

pB:wJ->sT 130

pB:wJ->sA 0

pB:wJ->sB 0

pC:wA->sA 0

pC:wA->sB 10

pC:wA->sC 0

pC:wA->sD 0

pC:wB->sC 30

pC:wB->sD 70

pC:wB->sE 30

pC:wB->sF 0

pC:wC->sE 0

pC:wC->sF 40

pC:wC->sG 0

pC:wC->sH 0

pC:wD->sG 50

pC:wD->sH 60

pC:wD->sI 0

pC:wD->sJ 110

pC:wE->sI 40

pC:wE->sJ 0

pC:wE->sK 90

pC:wE->sL 40

pC:wF->sK 10

pC:wF->sL 0

pC:wF->sM 0

pC:wF->sN 0

pC:wG->sM 30

pC:wG->sN 50

pC:wG->sO 0

pC:wG->sP 0

pC:wH->sO 60

pC:wH->sP 20

pC:wH->sQ 0

pC:wH->sR 0

pC:wI->sQ 30

pC:wI->sR 40

pC:wI->sS 0

pC:wI->sT 0

pC:wJ->sS 60

pC:wJ->sT 70

pC:wJ->sA 80

pC:wJ->sB 50

pD:wA->sA 0

pD:wA->sB 0

pD:wA->sC 0

pD:wA->sD 0

pD:wB->sC 20

pD:wB->sD 40

pD:wB->sE 0

pD:wB->sF 0

pD:wC->sE 10

pD:wC->sF 20

pD:wC->sG 0

pD:wC->sH 0

pD:wD->sG 30

pD:wD->sH 40

pD:wD->sI 0

pD:wD->sJ 0

pD:wE->sI 20

pD:wE->sJ 50

pD:wE->sK 50

pD:wE->sL 20

pD:wF->sK 0

pD:wF->sL 0

pD:wF->sM 0

pD:wF->sN 0

pD:wG->sM 30

pD:wG->sN 20

pD:wG->sO 0

pD:wG->sP 0

pD:wH->sO 40

pD:wH->sP 10

pD:wH->sQ 0

pD:wH->sR 0

pD:wI->sQ 20

pD:wI->sR 20

pD:wI->sS 0

pD:wI->sT 0

pD:wJ->sS 10

pD:wJ->sT 60

pD:wJ->sA 40

pD:wJ->sB 30

pE:wA->sA 220

pE:wA->sB 200

pE:wA->sC 0

pE:wA->sD 0

pE:wB->sC 150

pE:wB->sD 300

pE:wB->sE 0

pE:wB->sF 0

pE:wC->sE 140

pE:wC->sF 150

pE:wC->sG 240

pE:wC->sH 0

pE:wD->sG 0

pE:wD->sH 210

pE:wD->sI 0

pE:wD->sJ 0

pE:wE->sI 180

pE:wE->sJ 390

pE:wE->sK 360

`

pE:wF->sK 0

pE:wF->sL 0

pE:wF->sM 0

pE:wF->sN 0

pE:wG->sM 140

pE:wG->sN 200

pE:wG->sO 0

pE:wG->sP 0

pE:wH->sO 250

pE:wH->sP 130

pE:wH->sQ 0

pE:wH->sR 0

pE:wI->sQ 130

pE:wI->sR 100

pE:wI->sS 0

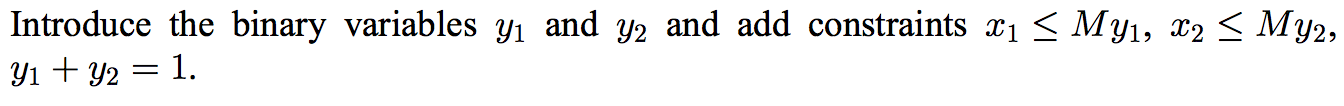
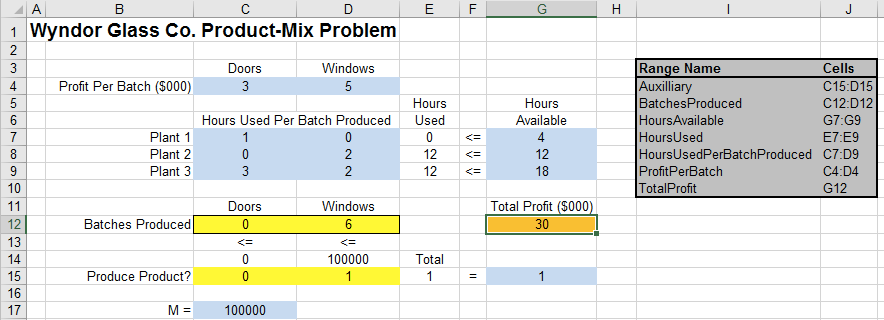
pE:wI->sT 0

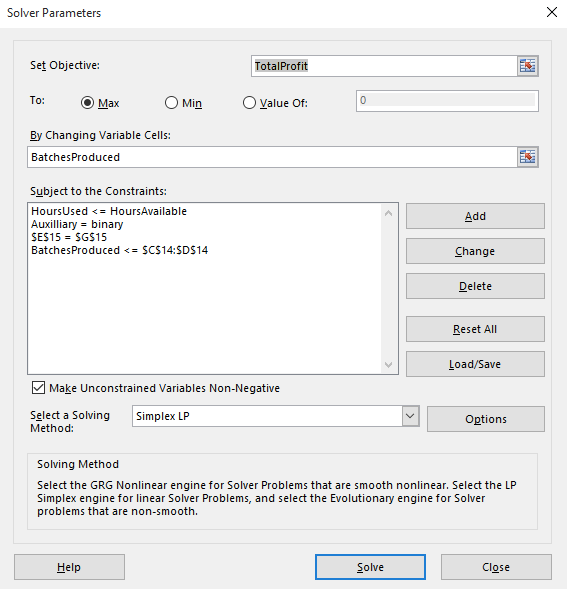
pE:wJ->sS 210

pE:wJ->sT 230

pE:wJ->sA 80

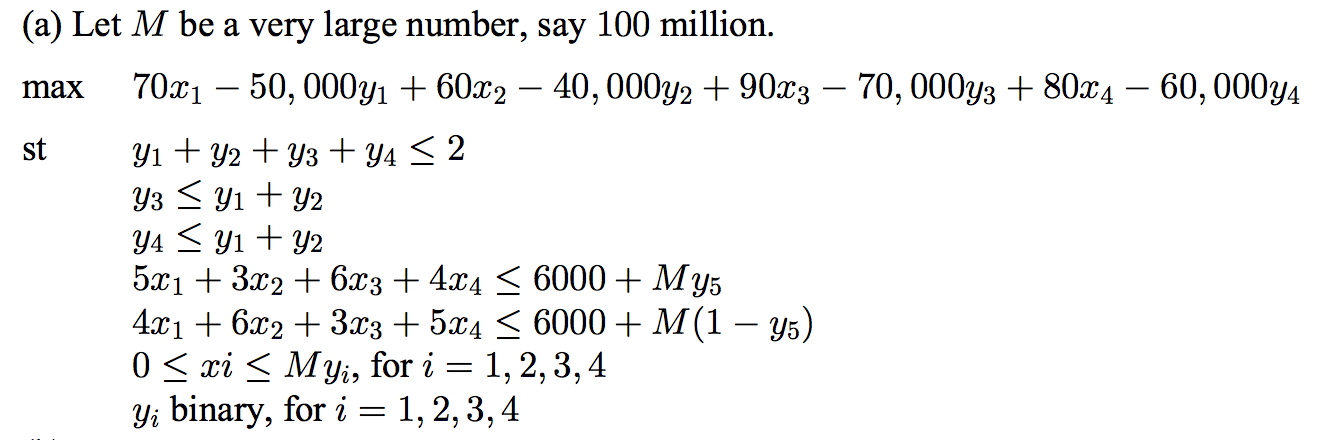
pE:wJ->sB 0

1. Problem 2: (12 pts) Complete Problem 12.4-2 from the book using Excel. Include a picture of your spreadsheet (or cut and paste it directly if possible) into your homework submission document.  



1. Problem 3: (12 pts) Complete Problem 12.3-1 from the book in OPL. Include your OPL files and the results in your write up. (You may have to read 12.3 and 12.4 carefully to learn how to use binary variables to implement the constraints.

You’re not required to report the formulated model, but here it is:



Here is the source code for solving the problem in OPL:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* OPL 12.6.3.0 Model

\* Author: jbaggett

\* Creation Date: Sep 22, 2016 at 8:17:05 PM

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//

float StartUpCost[1..4] = [50000, 40000, 70000, 60000];

float MarginalRevenue[1..4] = [70,60,90,80];

float ctCoef[1..2][1..4] = [ [5,3,6,4], [4,6,3,5] ];

float ctRhs[1..2] = [6000,6000];

float M = 100000000;

dvar float+ units[1..4];

dvar boolean product[1..4];

dvar boolean ctChoose;

maximize sum(j in 1..4) (MarginalRevenue[j]\*units[j] - StartUpCost[j]\*product[j]);

subject to{

ct1:

sum(j in 1..4) product[j] <= 2;

ct2:

product[3] <= product[1] + product[2];

ct3:

product[4] <= product[1] + product[2];

ct4:

sum(j in 1..4) ctCoef[1][j]\*units[j] <= ctRhs[1] + M\*ctChoose;

ct5:

sum(j in 1..4) ctCoef[2][j]\*units[j] <= ctRhs[2] + M\*(1-ctChoose);

ct6:

forall(j in 1..4)

units[j] <= M\*product[j];

};

The solution is to make 2000 units of product 2 for a maximum profit of 80,000.