crud

October 31, 2022

[ ]:
```
using JuMP
#Import HiGHS solver
using HiGHS

#Create a JuMP model named picframe1 that will be solved using the HiGHS solver
crud=Model(HiGHS.Optimizer);
#index for general time and time except last hour
time=[:10,:11,:12,:13,:14,:15];
time2=[:10,:11,:12,:13,:14];

#price factor
production=Dict(zip(time,[300,240,600,200,300,600]));
recy=Dict(zip(time,[30,40,35,45,38,50]));

#upper bound
storeBound=1000;
transBound=650;

#set variable
@variable(crud,stored[time]>=0);
@variable(crud,sent[time]>=0);

#set bound for requirement
@constraint(crud,storeConstarint[i in time],stored[i]<=storeBound);
@constraint(crud,sentConstarint[i in time],sent[i]<=transBound);

#set bound for the last hour. (The production for last hour+stored waste has to␣
 ↪less than
#  transportation upper bound )
@constraint(crud,lastHourStoreBound,stored[:15]<=transBound-production[:15]);

#dynamically update the stored variable except last index;
@constraint(crud,storeUpdate[i in␣
 ↪time2],stored[i+1]==stored[i]-sent[i]+production[i]);

#set the sent amount for last hour
@constraint(crud,LastSent,sent[:15]==stored[:15]+production[:15]);
```

```
#objective with cost
@objective(crud,Min,sum(sent[i]*recy[i] for i in time));


print(crud);
```

```
Min 30 sent[10] + 40 sent[11] + 35 sent[12] + 45 sent[13] + 38 sent[14] + 50
sent[15]
Subject to
 storeUpdate[10] : -stored[10] + stored[11] + sent[10] = 300.0
 storeUpdate[11] : -stored[11] + stored[12] + sent[11] = 240.0
 storeUpdate[12] : -stored[12] + stored[13] + sent[12] = 600.0
 storeUpdate[13] : -stored[13] + stored[14] + sent[13] = 200.0
 storeUpdate[14] : -stored[14] + stored[15] + sent[14] = 300.0
 LastSent : -stored[15] + sent[15] = 600.0
 storeConstarint[10] : stored[10]   1000.0
 storeConstarint[11] : stored[11]   1000.0
 storeConstarint[12] : stored[12]   1000.0
 storeConstarint[13] : stored[13]   1000.0
 storeConstarint[14] : stored[14]   1000.0
 storeConstarint[15] : stored[15]   1000.0
 sentConstarint[10] : sent[10]   650.0
 sentConstarint[11] : sent[11]   650.0
 sentConstarint[12] : sent[12]   650.0
 sentConstarint[13] : sent[13]   650.0
 sentConstarint[14] : sent[14]   650.0
 sentConstarint[15] : sent[15]   650.0
 lastHourStoreBound : stored[15]   50.0
 stored[10]   0.0
 stored[11]   0.0
 stored[12]   0.0
 stored[13]   0.0
 stored[14]   0.0
 stored[15]   0.0
 sent[10]   0.0
 sent[11]   0.0
 sent[12]   0.0
 sent[13]   0.0
 sent[14]   0.0
 sent[15]   0.0
```

[ ]:
```
optimize!(crud);
@show objective_value(crud);
@show value.(sent);
```

```
Presolving model
5 rows, 11 cols, 15 nonzeros
```

```
5 rows, 11 cols, 15 nonzeros
Presolve : Reductions: rows 5(-14); columns 11(-1); elements 15(-15)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     3.0000000000e+04 Pr: 5(1640) 0s
          7     8.8050000000e+04 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 7
Objective value     :  8.8050000000e+04
HiGHS run time      :          0.00
objective_value(crud) = 88050.0
value.(sent) = 1-dimensional DenseAxisArray{Float64,1,…} with index sets:
    Dimension 1, [10, 11, 12, 13, 14, 15]
And data, a 6-element Vector{Float64}:
 300.0
  40.0
 650.0
   0.0
 650.0
 600.0
```

salaries

October 31, 2022

[ ]:
```
#Import JuMP package to build an optimization model
using JuMP
#Import HiGHS solver
using HiGHS

#Create a JuMP model named picframe1 that will be solved using the HiGHS solver
picframe1 = Model(HiGHS.Optimizer);

#Add the variables
@variable(picframe1,tom>= 0);
@variable(picframe1,peter>=0);
@variable(picframe1,nina>=0);
@variable(picframe1,samir>=0);
@variable(picframe1,gary>=0);
@variable(picframe1,bob>=0);
@variable(picframe1,linda>=0);
@variable(picframe1,IT>=0);
@variable(picframe1,Customer>=0);

#Add constraint

@constraint(picframe1, constarint1, tom>=30000);
@constraint(picframe1, constarint2, nina>=tom+8000);
@constraint(picframe1, constarint3, peter>=tom+8000);
@constraint(picframe1, constarint4, samir>=tom+8000);
@constraint(picframe1, constarint5, gary>=tom+peter);
@constraint(picframe1, constarint6, linda==500+gary);
@constraint(picframe1, constarint7, nina+samir>=2*(tom+peter));
@constraint(picframe1, constarint8, bob>=peter);
@constraint(picframe1, constarint9, bob>=samir);
@constraint(picframe1, constarint10, bob+peter>=75000);
@constraint(picframe1, constarint11, linda<=bob+tom);

#convert problem to convex
@constraint(picframe1,constarint12,IT>=tom);
@constraint(picframe1,constarint13,IT>=peter);
@constraint(picframe1,constarint14,IT>=nina);
```

```
@constraint(picframe1,constarint15,IT>=samir);
@constraint(picframe1,constarint16,Customer>=gary);
@constraint(picframe1,constarint17,Customer>=bob);
@constraint(picframe1,constarint18,Customer>=linda);

#objective function
@objective(picframe1,Min,IT+Customer);



print(picframe1);
```

```
Min IT + Customer
Subject to
 constarint6 : -gary + linda = 500.0
 constarint1 : tom  30000.0
 constarint2 : -tom + nina  8000.0
 constarint3 : -tom + peter  8000.0
 constarint4 : -tom + samir  8000.0
 constarint5 : -tom - peter + gary  0.0
 constarint7 : -2 tom - 2 peter + nina + samir  0.0
 constarint8 : -peter + bob  0.0
 constarint9 : -samir + bob  0.0
 constarint10 : peter + bob  75000.0
 constarint12 : -tom + IT  0.0
 constarint13 : -peter + IT  0.0
 constarint14 : -nina + IT  0.0
 constarint15 : -samir + IT  0.0
 constarint16 : -gary + Customer  0.0
 constarint17 : -bob + Customer  0.0
 constarint18 : -linda + Customer  0.0
 constarint11 : -tom - bob + linda  0.0
 tom  0.0
 peter  0.0
 nina  0.0
 samir  0.0
 gary  0.0
 bob  0.0
 linda  0.0
 IT  0.0
 Customer  0.0
```

[ ]:
```
optimize!(picframe1);
@show objective_value(picframe1);
@show value(Customer);
@show value(IT);
```

```
Presolving model
```

```
16 rows, 8 cols, 36 nonzeros
13 rows, 5 cols, 30 nonzeros
6 rows, 4 cols, 13 nonzeros
3 rows, 3 cols, 6 nonzeros
Presolve : Reductions: rows 3(-15); columns 3(-6); elements 6(-33)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     1.0650020661e+05 Pr: 1(60000) 0s
          2     1.3650000000e+05 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 2
Objective value     :  1.3650000000e+05
HiGHS run time      :          0.00
objective_value(picframe1) = 136500.0
value(Customer) = 68500.0
value(IT) = 68000.0
```

whiskey

October 31, 2022

[ ]:
```
using JuMP
#Import HiGHS solver
using HiGHS

#Create a JuMP model named picframe1 that will be solved using the HiGHS solver
wiskey=Model(HiGHS.Optimizer);
wistype = [:std,:cho,:pri];
blends = [:scot,:johnny];
wisAvilable=Dict(zip(wistype,[1200,2500,2000]));
wisCost=Dict(zip(wistype,[4,5,7]));
demand=Dict(zip(blends,[1000,600]));
wisPrice=Dict(zip(blends,[6.8,5.7]));

#variable for money spend on advertise and quantity used for each type
@variable(wiskey,x[wistype,blends]>=0);
@variable(wiskey,money[blends]>=0);

#Sup and Inf for specific type of liquor
@constraint(wiskey,PSInf,x[:pri,:scot]>=0.6*sum(x[i,:scot] for i in wistype));
@constraint(wiskey,SCSup,x[:std,:scot]<=0.2*sum(x[i,:scot] for i in wistype));
@constraint(wiskey,PJInf,x[:pri,:johnny]>=0.15*sum(x[i,:johnny] for i in␣
 ↪wistype));
@constraint(wiskey,SJSup,x[:std,:johnny]<=0.6*sum(x[i,:johnny] for i in␣
 ↪wistype));

#selling constant with advertise
@constraint(wiskey,prodcutInf[j in blends],sum(x[i,j] for i in␣
 ↪wistype)<=demand[j]+1.25*money[j]);

#avilable liquor
@constraint(wiskey, WisConstraint[i in wistype], sum(x[i,j] for j in␣
 ↪blends)<=wisAvilable[i]);

#Objetive with revenue-cost-money spend on advertise
@objective(wiskey,Max, sum(x[i,j] for i in wistype for j in blends)*wisPrice[j]-
sum(x[i,j] for i in wistype for j in blends)*wisCost[i]-sum(money[i] for i in␣
 ↪blends));
```

```
print(wiskey);
```

```
Max 2.8 x[std,scot] + 1.7000000000000002 x[std,johnny] + 1.7999999999999998
x[cho,scot] + 0.7000000000000002 x[cho,johnny] - 0.20000000000000018 x[pri,scot]
- 1.2999999999999998 x[pri,johnny] - money[scot] - money[johnny]
Subject to
 PSInf : -0.6 x[std,scot] - 0.6 x[cho,scot] + 0.4 x[pri,scot]  0.0
 PJInf : -0.15 x[std,johnny] - 0.15 x[cho,johnny] + 0.85 x[pri,johnny]  0.0
 SCSup : 0.8 x[std,scot] - 0.2 x[cho,scot] - 0.2 x[pri,scot]  0.0
 SJSup : 0.4 x[std,johnny] - 0.6 x[cho,johnny] - 0.6 x[pri,johnny]  0.0
 prodcutInf[scot] : x[std,scot] + x[cho,scot] + x[pri,scot] - 1.25 money[scot]
1000.0
 prodcutInf[johnny] : x[std,johnny] + x[cho,johnny] + x[pri,johnny] - 1.25
money[johnny]  600.0
 WisConstraint[std] : x[std,scot] + x[std,johnny]  1200.0
 WisConstraint[cho] : x[cho,scot] + x[cho,johnny]  2500.0
 WisConstraint[pri] : x[pri,scot] + x[pri,johnny]  2000.0
 x[std,scot]  0.0
 x[cho,scot]  0.0
 x[pri,scot]  0.0
 x[std,johnny]  0.0
 x[cho,johnny]  0.0
 x[pri,johnny]  0.0
 money[scot]  0.0
 money[johnny]  0.0
```

[ ]:
```
optimize!(wiskey)
@show objective_value(wiskey)
@show value.(x);
```

```
Presolving model
9 rows, 8 cols, 26 nonzeros
9 rows, 8 cols, 26 nonzeros
Presolve : Reductions: rows 9(-0); columns 8(-0); elements 26(-0)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0    -1.1199987222e+01 Ph1: 7(10.8); Du: 4(11.2) 0s
          7    -1.6133333333e+03 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 7
Objective value     :  1.6133333333e+03
HiGHS run time      :          0.00
objective_value(wiskey) = 1613.3333333333335
value.(x) = 2-dimensional DenseAxisArray{Float64,2,…} with index sets:
```

```
    Dimension 1, [:std, :cho, :pri]
    Dimension 2, [:scot, :johnny]
And data, a 3×2 Matrix{Float64}:
 199.99999999999997  1000.0
 200.00000000000006   416.66666666666674
 600.0                250.0
```