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
```