9

October 2, 2023

[ ]:

```
using JuMP
using CPLEX
```

[ ]:

```
order_details = [14 5 200; 31 10 350; 36 15 400; 45 5 500]
```

```
4×3 Matrix{Int64}:
 14   5  200
 31  10  350
 36  15  400
 45   5  500
```

[ ]:

```
scrap_price = 5
```

```
5
```

[ ]:

```
manufacturing_cost = 700
```

```
700
```

[ ]:

```
model = Model(CPLEX.Optimizer)
```

```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: CPLEX
```

[ ]:

```
@variable(model, x[i=1:4, j =1:10], lower_bound = 0, Int) # Amount of orders of each type to be cut from available 10 rolls
```

```
4×10 Matrix{VariableRef}:
 x[1,1]  x[1,2]  x[1,3]  x[1,4]  x[1,5]  …  x[1,7]  x[1,8]  x[1,9]  x[1,10]
 x[2,1]  x[2,2]  x[2,3]  x[2,4]  x[2,5]     x[2,7]  x[2,8]  x[2,9]  x[2,10]
 x[3,1]  x[3,2]  x[3,3]  x[3,4]  x[3,5]     x[3,7]  x[3,8]  x[3,9]  x[3,10]
 x[4,1]  x[4,2]  x[4,3]  x[4,4]  x[4,5]     x[4,7]  x[4,8]  x[4,9]  x[4,10]
```

[ ]:

```
@constraint(model, sum(x[:, j] for j in 1:10) <= order_details[:, 2]) # Constraint to make sure that production of any particular type do not exceedits demand
```

$[x_{1,1}+x_{1,2}+x_{1,3}+x_{1,4}+x_{1,5}+x_{1,6}+x_{1,7}+x_{1,8}+x_{1,9}+x_{1,10}-5, x_{2,1}+x_{2,2}+x_{2,3}+x_{2,4}+x_{2,5}+x_{2,6}+x_{2,7}+x_{2,8}+x_{2,9}+x_{2,10}$

[ ]:

```
@constraint(model, sum((x .*order_details[:, 1])[i,:] for i in 1:4) .<= 100) #␣
  ↪Roll length contraint
```

```
10-element Vector{ConstraintRef{Model, MathOptInterface.
 ↪ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64},␣
 ↪MathOptInterface.LessThan{Float64}}, ScalarShape}}:
 14 x[1,1] + 31 x[2,1] + 36 x[3,1] + 45 x[4,1] <= 100
 14 x[1,2] + 31 x[2,2] + 36 x[3,2] + 45 x[4,2] <= 100
 14 x[1,3] + 31 x[2,3] + 36 x[3,3] + 45 x[4,3] <= 100
 14 x[1,4] + 31 x[2,4] + 36 x[3,4] + 45 x[4,4] <= 100
 14 x[1,5] + 31 x[2,5] + 36 x[3,5] + 45 x[4,5] <= 100
 14 x[1,6] + 31 x[2,6] + 36 x[3,6] + 45 x[4,6] <= 100
 14 x[1,7] + 31 x[2,7] + 36 x[3,7] + 45 x[4,7] <= 100
 14 x[1,8] + 31 x[2,8] + 36 x[3,8] + 45 x[4,8] <= 100
 14 x[1,9] + 31 x[2,9] + 36 x[3,9] + 45 x[4,9] <= 100
 14 x[1,10] + 31 x[2,10] + 36 x[3,10] + 45 x[4,10] <= 100
```

[ ]:

```
scrap = 100 .- sum((x .*order_details[:, 1])[i,:] for i in 1:4)
```

```
10-element Vector{AffExpr}:
 -14 x[1,1] - 31 x[2,1] - 36 x[3,1] - 45 x[4,1] + 100
 -14 x[1,2] - 31 x[2,2] - 36 x[3,2] - 45 x[4,2] + 100
 -14 x[1,3] - 31 x[2,3] - 36 x[3,3] - 45 x[4,3] + 100
 -14 x[1,4] - 31 x[2,4] - 36 x[3,4] - 45 x[4,4] + 100
 -14 x[1,5] - 31 x[2,5] - 36 x[3,5] - 45 x[4,5] + 100
 -14 x[1,6] - 31 x[2,6] - 36 x[3,6] - 45 x[4,6] + 100
 -14 x[1,7] - 31 x[2,7] - 36 x[3,7] - 45 x[4,7] + 100
 -14 x[1,8] - 31 x[2,8] - 36 x[3,8] - 45 x[4,8] + 100
 -14 x[1,9] - 31 x[2,9] - 36 x[3,9] - 45 x[4,9] + 100
 -14 x[1,10] - 31 x[2,10] - 36 x[3,10] - 45 x[4,10] + 100
```

[ ]:

```
profit = sum(sum((x .* order_details[:, 3])[i,:] for i in 1:4)) + sum(scrap *␣
  ↪scrap_price) - manufacturing_cost*10
```

$130x_{1,1}+195x_{2,1}+220x_{3,1}+275x_{4,1}+130x_{1,2}+195x_{2,2}+220x_{3,2}+275x_{4,2}+130x_{1,3}+195x_{2,3}+220x_{3,3}+275x_{4,3}+130x_1$

[ ]:

```
@objective(model, Max, profit)
```

$130x_{1,1}+195x_{2,1}+220x_{3,1}+275x_{4,1}+130x_{1,2}+195x_{2,2}+220x_{3,2}+275x_{4,2}+130x_{1,3}+195x_{2,3}+220x_{3,3}+275x_{4,3}+130x_1$

[ ]:

```
optimize!(model)
```

```
Mixed integer rounding cuts applied:  41
Zero-half cuts applied:  2

Root node processing (before b&c):
  Real time             =    0.00 sec. (0.01 ticks)
Parallel b&c, 8 threads:
  Real time             =    9.36 sec. (2016.17 ticks)
  Sync time (average)   =    1.29 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    9.36 sec. (2016.17 ticks)
Version identifier: 22.1.1.0 | 2022-11-26 | 9160aff4d
Found incumbent of value -2000.000000 after 0.00 sec. (0.00 ticks)
Tried aggregator 1 time.
Reduced MIP has 14 rows, 40 columns, and 80 nonzeros.
Reduced MIP has 0 binaries, 40 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (0.04 ticks)
Tried aggregator 1 time.
Detecting symmetries...
Reduced MIP has 14 rows, 40 columns, and 80 nonzeros.
Reduced MIP has 0 binaries, 40 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (0.07 ticks)
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 8 threads.
Root relaxation solution time = 0.00 sec. (0.06 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                       -2000.0000    20250.0000             ---
*     0+    0                        4175.0000    20250.0000           385.03%
      0     0     4388.8889    10     4175.0000     4388.8889       28    5.12%
      0     0     4388.8889    18     4175.0000      Cuts: 13       52    5.12%
      0     0     4388.8889    19     4175.0000      Cuts: 16       79    5.12%
      0     2     4388.8889    19     4175.0000     4388.8889       79    5.12%
Elapsed time = 0.06 sec. (1.11 ticks, tree = 0.02 MB, solutions = 2)
```

```
[ ]: @show value.(x) # Amount of rolls of each type to be cut to maximise profit on␣
       ↪each available roll
```

```
value.(x) = [1.0 0.0 2.0 0.0 2.0 0.0 0.0 0.0 0.0 0.0; 0.0 2.0 0.0 2.0 0.0 2.0
0.0 2.0 0.0 2.0; 1.0 1.0 2.0 1.0 2.0 1.0 0.0 1.0 0.0 1.0; 1.0 0.0 0.0 0.0 0.0
0.0 2.0 0.0 2.0 0.0]

4×10 Matrix{Float64}:
 1.0  0.0  2.0  0.0  2.0  0.0  0.0  0.0  0.0  0.0
 0.0  2.0  0.0  2.0  0.0  2.0  0.0  2.0  0.0  2.0
 1.0  1.0  2.0  1.0  2.0  1.0  0.0  1.0  0.0  1.0
 1.0  0.0  0.0  0.0  0.0  0.0  2.0  0.0  2.0  0.0
```

```
[ ]: @show objective_value(model)
```

```
objective_value(model) = 4175.0
```

```
4175.0
```