Challenge Decembre 2025

Inside/Outside Production

A solution with OPL CPLEX by Alex Fleischer
afleischer@fr.ibm.com

OPL (Optimization Programming Language) is an abstract modeling language that helps model easily optimization problems that can be solved both with IBM CPLEX linear programming and IBM CPLEX constraint programming CPOptimizer (CPO)

This can work on your machine or in the cloud through IBM watsonx AI

“You need to help a manufacturer decide how much of each demanded product should be produced internally and how much should be sourced from outside. Whether a product is made inside or outside, it has an associated cost. A product can consume a given amount of internal resources that have limited capacities. The general objective is to minimize the total production cost while ensuring the company meets the demand exactly or with a certain tolerance.

The manufacturer wants to consider various production decisions to choose the most suitable ones based on the long-term resource availability and uncertain future demand.”

This is clearly an optimization problem that is good for CPLEX and we can find a model for that in the examples within the product.

The example is at examples/opl/production

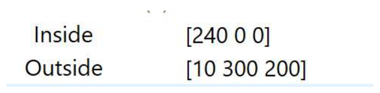
The model:

```
// --------------------------------------------------------------------------
// Licensed Materials - Property of IBM
//
// 5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55
// Copyright IBM Corporation 1998, 2024. All Rights Reserved.
//
// Note to U.S. Government Users Restricted Rights:
// Use, duplication or disclosure restricted by GSA ADP Schedule
// Contract with IBM Corp.
// --------------------------------------------------------------------------

{string} Products = ...;
{string} Resources = ...;

float Consumption[Products][Resources] = ...;
float Capacity[Resources] = ...;
float Demand[Products] = ...;
float InsideCost[Products] = ...;
float OutsideCost[Products]  = ...;

dvar float+ Inside[Products];
dvar float+ Outside[Products];

minimize
  sum( p in Products )
    ( InsideCost[p] * Inside[p] + OutsideCost[p] * Outside[p] );

subject to {
  forall( r in Resources )
    ctCapacity:
      sum( p in Products )
        Consumption[p][r] * Inside[p] <= Capacity[r];

  forall(p in Products)
    ctDemand:
      Inside[p] + Outside[p] >= Demand[p];
}
```

Now I need to use the right data to comply with the challenge:

```
Products =  { "P1", "P2", "P3" };
Resources = { "R1", "R2","R3" };

Consumption = [ [5, 2, 0], [4, 4, 0], [3, 0, 6] ];
Capacity = [ 1200, 800,700 ];
Demand = [ 250, 300, 200 ];
InsideCost = [ 6, 8, 3 ];
OutsideCost  = [ 8, 9, 4 ];
```

And when I run the model I get



Inside	[240 0 0]
Outside	[10 300 200]

And

```
// solution (optimal) with objective 5020
```

Today the world is getting uncertain and users tend to prefer to be able to choose among several very good solutions. So instead of asking for one solution only, let’s ask for several solutions and for that we can rely on solution pools.

```
 {string} Products = ...;
{string} Resources = ...;

float Consumption[Products][Resources] = ...;
float Capacity[Resources] = ...;
float Demand[Products] = ...;
float InsideCost[Products] = ...;
float OutsideCost[Products]  = ...;

dvar int+ Inside[Products];
dvar int+ Outside[Products];

minimize
  sum( p in Products )
    ( InsideCost[p] * Inside[p] + OutsideCost[p] * Outside[p] );

subject to {
  forall( r in Resources )
    ctCapacity:
      sum( p in Products )
        Consumption[p][r] * Inside[p] <= Capacity[r];

  forall(p in Products)
    ctDemand:
      Inside[p] + Outside[p] >= Demand[p];
}

execute
{
  writeln("objective = ",cplex.getObjValue());
  writeln("Inside = ",Inside);
  writeln("Outside = ",Outside);
  writeln();
}

main

{

   //writeln("solution pools")

   thisOplModel.generate();
   cplex.solnpoolintensity=4;
   cplex.solnpoolcapacity=20;
   cplex.solnpoolreplace=2;
   cplex.populatelim=20;


   cplex.populate();

   var nsolns = cplex.solnPoolNsolns;
   writeln("n sol ",nsolns);



   for(var s = 0; s < nsolns; s++)
   {
      thisOplModel.setPoolSolution(s);
      thisOplModel.postProcess();


   }
}
```

With the same data gives

```
n sol 20
objective = 5020
Inside =  [240 0 0]
Outside =  [10 300 200]

objective = 5054
Inside =  [240 0 0]
Outside =  [12 302 200]

objective = 5058
Inside =  [240 0 0]
Outside =  [12 302 201]

objective = 5072
Inside =  [240 0 0]
Outside =  [12 304 200]

objective = 5099
Inside =  [240 0 0]
Outside =  [12 307 200]

objective = 5025
Inside =  [239 1 0]
Outside =  [11 299 201]

objective = 5037
Inside =  [239 1 0]
Outside =  [13 299 200]

objective = 5053
Inside =  [239 1 0]
Outside =  [15 299 200]

objective = 5061
Inside =  [239 1 0]
Outside =  [16 299 200]
```

```
objective = 5023
Inside =  [237 3 0]
Outside =  [13 297 200]

objective = 5033
Inside =  [236 4 0]
Outside =  [14 297 200]

objective = 5025
Inside =  [236 3 0]
Outside =  [14 297 200]

objective = 5031
Inside =  [237 3 0]
Outside =  [14 297 200]

objective = 5027
Inside =  [237 3 0]
Outside =  [13 297 201]

objective = 5024
Inside =  [236 4 0]
Outside =  [14 296 200]

objective = 5033
Inside =  [235 6 0]
Outside =  [15 295 200]

objective = 5025
Inside =  [235 5 0]
Outside =  [15 295 200]

objective = 5031
Inside =  [236 5 0]
Outside =  [15 295 200]

objective = 5027
Inside =  [236 5 0]
Outside =  [14 295 201]

objective = 5042
Inside =  [237 0 0]
Outside =  [15 300 200]
```