Updated on March , 2020

IBM ILOG CPLEX Optimization Studio

Prakash Kotecha (Associate Professor)
Debasis Maharana, Teaching Assistant
Remya Kommadath, Teaching Assistant
Indian Institute of Technology Guwahati

IBM ILOG CPLEX Optimization Studio:
Constraint Programming Applications in IBM ILOG CPLEX Optimization Studio:
Additional resources: tinyurl.com/sksopti, tinyurl.com/sksoptivid

IBM ILOG CPLEX Optimization Studio

- A consolidation of Optimization Programming Language (OPL) integrated development environment , CPLEX and CP Optimizer solution engines.
- **CPLEX Optimization Studio** provides a faster way to build efficient optimization models.
- Total separation between model and data helps in testing multiple data on single model
- Consists of
  - An integrated development environment (IDE)
  - A mathematical optimization engine (CPLEX) for planning problems
  - A constraint programming engine (CP) for scheduling problems
  - A set of APIs (Python, Java, C#, etc.) for modeling, solving and embedding optimization solutions
- Supported on 64-bit windows, Linux and Mac OS
- Free version is available for students and faculties as part of IBM academic initiative.

https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html

IBM ILOG CPLEX Optimization Studio

IBM ILOG CPLEX Optimization Studio

Transform your business decision-making with data science

Nov 15, 2019

IBM ILOG CPLEX Optimization Studio uses decision optimization technology to optimize your business decisions, develop and deploy optimization models quickly, and create real-world applications that can significantly improve business outcomes. How? IBM ILOG CPLEX Optimization Studio is a prescriptive analytics solution that enables rapid development and deployment of decision optimization models using mathematical and constraint programming. It combines a fully featured integrated development environment that supports Optimization Programming Language (OPL) and the high-performance CPLEX and CP Optimizer solvers.

Products:

IBM ILOG CPLEX Optimization Studio

Videos

Watch technical experts walk you through common use cases, highlighting product features key capabilites.

Introduction—IBM ILOG CPLEX Optimization Studio

Solve a Production Planning problem using IBM ILOG CPLEX Optimization Studio IDE

Creating project within CPLEX Studio IDE

https://www.ibm.com/demos/collection/IBM-ILOG-CPLEX-Optimization-Studio/

Linear programming

- Minimize the cost of shipping goods from 2 canning plants to 3 markets, subject to supply and demand constraints.

- Details of distance between the plant and market (thousand miles), capacity of each plant and the demand of commodity in each market is given

Plants	Markets			Supply (cases)
	New York	Chicago	Topeka	
Seattle	2.5	1.7	1.8	350
San Diego	2.5	1.8	1.4	600
Demand (cases)	325	300	275	

- Freight in dollars per case per thousand miles is 90

Problem formulation

Parameters:

d_{ij}: distance between each plant and market

F: freight in dollars per case per thousand miles

a_i: supply of commodity in plant i (in cases)

b_i: demand for commodity at market j (in cases)

C_{ij}: cost per unit shipment between plant i and market j

$$C_{ij} = \frac{Fd_{ij}}{1000}$$

Decision variables: x_{ij} be the quantities of commodity transported from i^{th} plant to j^{th} market

Objective function: Minimize the total transportation costs (Z) in thousands of dollars

$$\boxed{Z = \sum_{i=1}^{2}\sum_{j=1}^{3} C_{ij} x_{ij}}$$

Problem formulation

Subject to

$$\sum_{j=1}^{3} x_{ij} \le a_i \quad \forall i \in \{1,2\}$$ → Supply constraint of each plant

$$\sum_{i=1}^{2} x_{ij} \ge b_j \quad \forall j \in \{1,2,3\}$$ → Demand constraint for each market

$$x_{ij} \ge 0 \quad \forall i \in \{1,2\}; \forall j \in \{1,2,3\}$$ → Bounds

Production planning data

SLPP_RK.mod SLPP_RK.dat

```
j = 54;
k = 2;

B = 1000;
R = [500,500];

SheetConnection Data("PPdata.xlsx");

l from SheetRead(Data,"sheet1!A3:A56");
m from SheetRead(Data,"sheet1!B3:B56");
h from SheetRead(Data,"sheet1!C3:C56");

cl  from SheetRead(Data,"sheet1!D3:D56");
cm  from SheetRead(Data,"sheet1!E3:E56");
ch  from SheetRead(Data,"sheet1!F3:F56");

il  from SheetRead(Data,"sheet1!G3:G56");
im  from SheetRead(Data,"sheet1!H3:H56");
ih  from SheetRead(Data,"sheet1!I3:I56");

rm from SheetRead(Data,"sheet1!J3:K56");


SP  =   [.975 ,.975,  .975,  .975,  .975,  .780
```

Defining the number of processes and raw material

Defining the available budget and raw materials

Reading the production capacity level, production cost, investment cost and raw materials required from the specified range of sheet 1

Defining the selling price of products produced by j processes

PPdata - Excel

	A	B	C	D	E	F	G	H	I	J	K
1	Capacity			Production Cost			Investment Cost			Raw material	
2	l	m	h	cl	cm	ch	il	im	ih	rm1	rm2
3	70	135	270	50.7	90.1	170.7	55	81.1	131.6	0.948	0
4	75	150	300	56.8	103.8	196.2	58	85.1	132.4	0.9432	0
5	77.5	155	310	56.9	103.7	195.7	60.2	86.8	134.1	0.949	0
6	70	145	290	51.7	97.6	184.8	55.1	83.1	132	0.9546	0
7	47.5	95	190	38.2	69.8	130.4	43.3	66.8	104.3	0.955	0
8	40	80	160	38.5	65.2	120.7	66.2	92.8	153.2	1.045	0
9	40	80	160	31.8	57.1	105.5	40	61.4	95.1	1.05	0
10	45	90	180	37.8	57.7	94.9	106.6	151.7	231.5	0.5103	0
11	40	80	160	38.5	65.6	119.1	82.8	125.4	207	0.6289	0
12	90	180	360	92.2	159.2	290.9	233.5	390.7	698.7	0.8648	0

Sheet1

Production planning model: Declaration of data

All the data are read from an external data file

*SLPP_RK.mod | SLPP_RK.dat

```
// Declaration of Data
int j = ...;
range J = 1 .. j;

int k = ...;
range K = 1..k;

float B = ...;
int R[K] = ...;

float SP[J] = ...;

float l[J] = ...;
float m[J] = ...;
float h[J] = ...;

float il[J] = ...;
float im[J] = ...;
float ih[J] = ...;

float cl[J] = ...;
float cm[J] = ...;
float ch[J] = ...;

float rm [J][K] = ...;
```


SLPP_RK.mod | SLPP_RK.dat

```


j = 54;
k = 2;

B = 1000;
R = [500,500];

SheetConnection Data("PPdata.xlsx");

l from SheetRead(Data,"sheet1!A3:A56");
m from SheetRead(Data,"sheet1!B3:B56");
h from SheetRead(Data,"sheet1!C3:C56");

cl from SheetRead(Data,"sheet1!D3:D56");
cm from SheetRead(Data,"sheet1!E3:E56");
ch from SheetRead(Data,"sheet1!F3:F56");

il from SheetRead(Data,"sheet1!G3:G56");
im from SheetRead(Data,"sheet1!H3:H56");
ih from SheetRead(Data,"sheet1!I3:I56");

rm from SheetRead(Data,"sheet1!J3:K56");


SP  =   [.975 ,.975, .975, .975, .975, .780,
```

Production planning model: Declaration of decision variables

SLPP_RK.mod   SLPP_RK.dat

```
// Declaration of Decision Variables

dvar boolean Y[J];

dvar boolean Z[J];

dvar float+ X[j in J];

dvar float+ L[j in J];

dvar float+ M[j in J];

dvar float+ H[j in J];
```


$$L_j \le Y_j \qquad \forall j = 1,2,\ldots,J$$

$$H_j \le 1 - Y_j \qquad \forall j = 1,2,\ldots,J$$

$$L_j + M_j + H_j = Z_j \qquad \forall j = 1,2,\ldots,J$$

$$X_j = l_j \cdot L_j + m_j \cdot M_j + h_j \cdot H_j \qquad \forall j = 1,2,\ldots,J$$

$$X_j \le U \cdot Z_j \qquad \forall j = 1,2,\ldots,J$$

$$Y_j, Z_j = 0 \; or \; 1 \qquad \forall j = 1,2,\ldots,J$$

$$X_j, L_j, M_j, H_j \ge 0 \qquad \forall j = 1,2,\ldots,J$$

Production planning model: Objective function and Constraints

*SLPP_RK.mod   SLPP_RK.dat

```
// Objective
maximize sum(j in J)(X[j]*SP[j])- sum(j in J)(cl[j]*L[j]+cm[j]*M[j]+ch[j]*H[j]);

subject to {

// Investment cost constraint
sum(j in J)(il[j]*L[j]+im[j]*M[j]+ih[j]*H[j])<=B;

//Raw material Constraint
forall (k in K){
sum(j in J)rm[j][k]*X[j]<= R[k];
}

// Production Capacity constraint
forall (j in J) {
L[j]<= Y[j];
H[j]<=1-Y[j];
L[j] + M[j] + H[j] == Z[j];
X[j] == l[j]*L[j] + m[j]*M[j] + h[j]*H[j];
X[j] <= 100000*Z[j];
}

}
```

$$\text{Max profit } = \sum_{j=1}^{J}\left(SP_j \cdot X_j - PC_j\right)$$

$$PC_j = cl_j \cdot L_j + cm_j \cdot M_j + ch_j \cdot H_j, \quad \forall j = 1,2,\dots,J$$

$$\sum_{j=1}^{J} il_j \cdot L_j + im_j \cdot M_j + ih_j \cdot H_j \le B$$

$$IC_j = il_j \cdot L_j + im_j \cdot M_j + ih_j \cdot H_j, \quad \forall j = 1,2,\dots,J$$

$$\sum_{j=1}^{J} rm_{jk} \cdot X_j \le R_k, \qquad k = 1,\dots,K$$

$$L_j \le Y_j \qquad \forall j = 1,2,\dots,J$$

$$H_j \le 1 - Y_j \qquad \forall j = 1,2,\dots,J$$

$$L_j + M_j + H_j = Z_j \qquad \forall j = 1,2,\dots,J$$

$$X_j = l_j \cdot L_j + m_j \cdot M_j + h_j \cdot H_j \qquad \forall j = 1,2,\dots,J$$

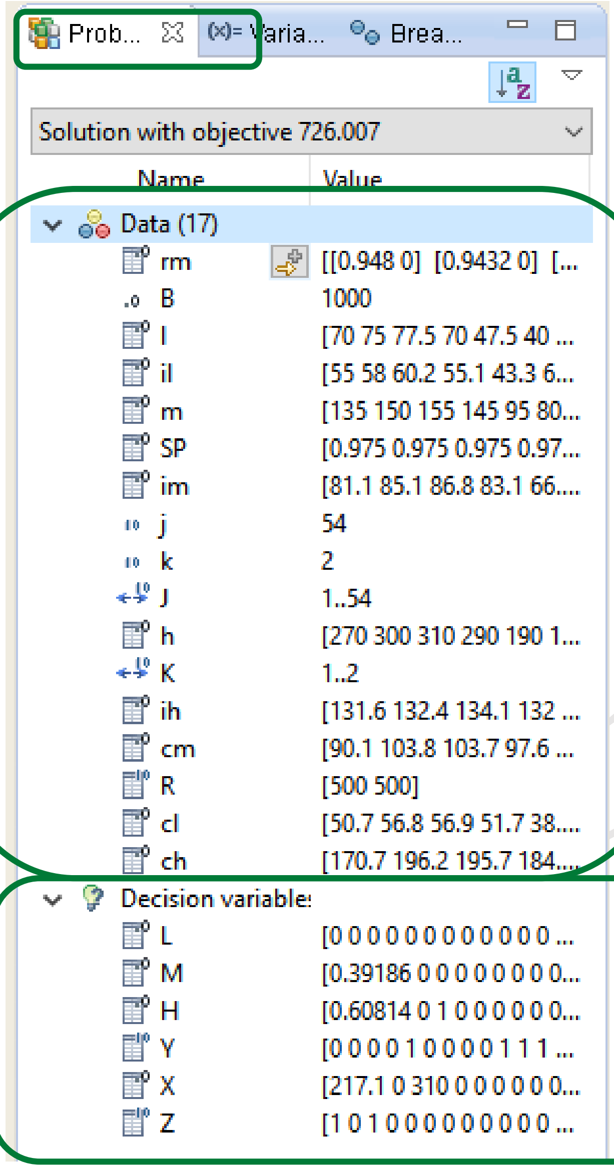
$$X_j \le U \cdot Z_j \qquad \forall j = 1,2,\dots,J$$

Result analysis

```
Problems  Scripting log  Solutions  Conflicts  Relaxations  Engine log  Statistics
// MILP solution norm |x| (Total, Max)            2.35854e+003 6.80000e+002
// MILP solution error (Ax=b) (Total, Max)        8.52651e-014 5.68434e-014
// MILP x bound error (Total, Max)                0.00000e+000 0.00000e+000
// MILP x integrality error (Total, Max)          0.00000e+000 0.00000e+000
// MILP slack bound error (Total, Max)            8.52651e-014 5.68434e-014
//

X = [217.1
         0 310 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
         0 0 0 0 0 0 0 50 0 0 0 0 613.44 0 680 450 0 0 0 0 0];
L = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
         0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
M = [0.39186 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
         0 0 0 0 0 0 0 0 0 0 0 0 0.28938 0 0 1 0 0 0 0 0];
H = [0.60814 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
         0 0 0 0 0 0 0 1 0 0 0 0 0.71062 0 1 0 0 0 0 0 0];
Y = [0 0 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0
         0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 1 0];
Z = [1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
         0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0];
```

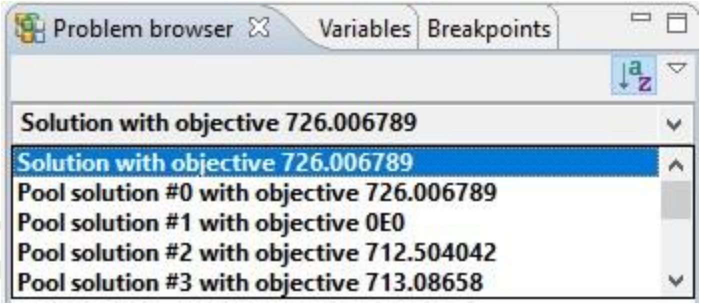
```
*SLPP_RK.mod  SLPP_RK.dat

// Display
execute{
for(var j in J)
if(X[j]>0)
writeln(j,'-',X[j]);
}
```

```
Problems  Scripting log  Solutions  Conflicts  Relaxations  Engine log  Statistics
Inter « Scripting log (drop script code here to execute it)
  // solution (integer optimal, tolerance) with objective 726.006789317354
  1-217.099156
  3-310
  41-50
  46-613.441716
  48-680
  49-450
```



Data provided

Obtained solution

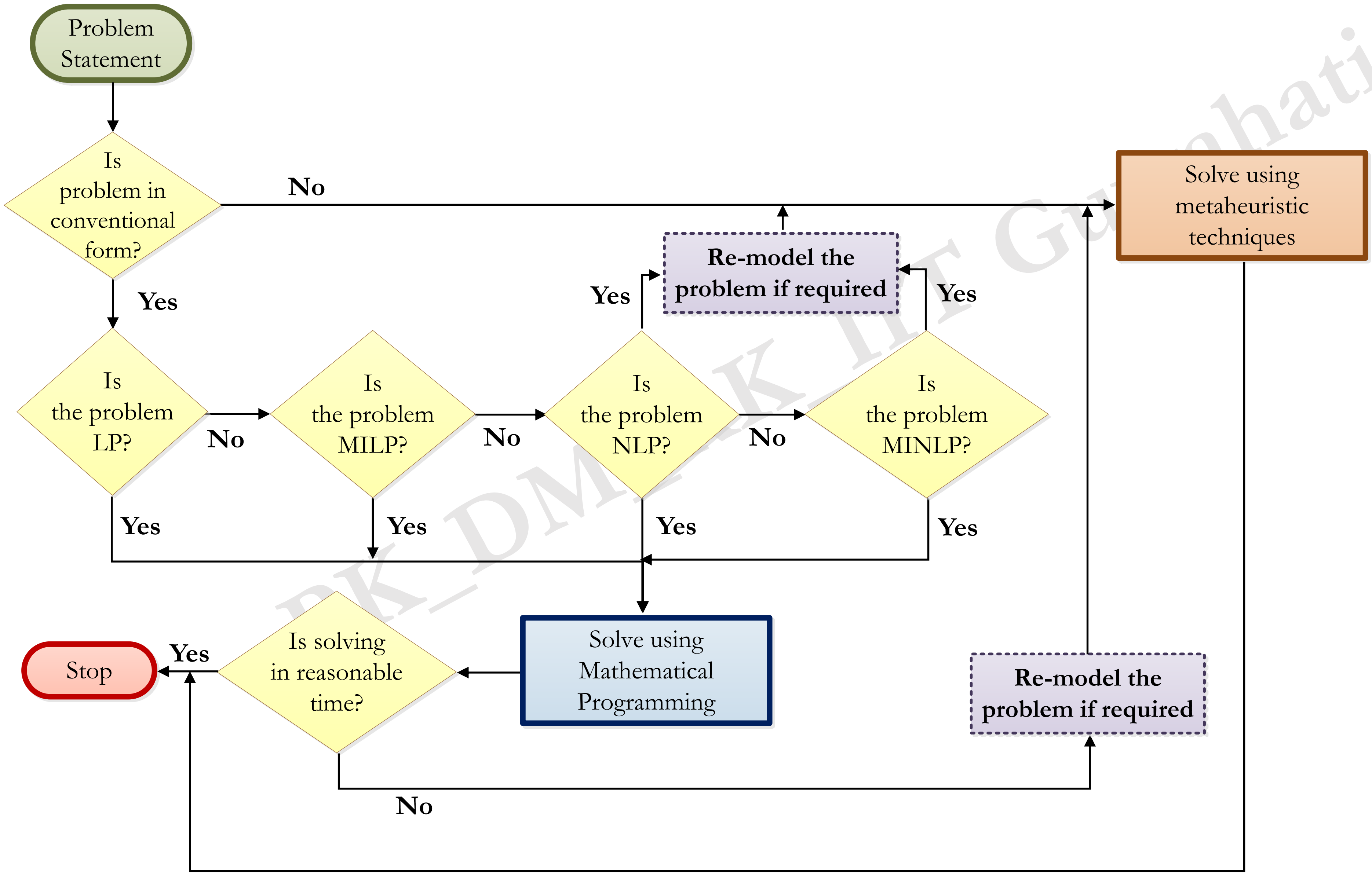


All integer feasible solutions are displayed in the drop down list

Result comparison of production planning

Resources [B, R1, R2]	GAMS		CPLEX	Metaheuristic Technique					
				Without correction			With correction		
	default optcr	optcr = 0.00001		TLBO	DE	PSO	TLBO	DE	PSO
[1000, 500, 500]	712.50	726.01	726.01	400.59	1.17E+20	546.28	699.38	690.82	710.04
[1000, 1000, 1000]	834.30	834.30	834.30	622.51	2.00E+20	639.80	790.78	816.72	750.45
[2000, 500, 500]	1133.15	1173.11	1173.11	757.77	419.19	647.70	1066.3	1092.3	857.91
[2000, 1000, 1000]	1452.82	1452.82	1452.82	1077.50	463.96	922.38	1360.27	1375.50	1297.05

Selection procedure



$$\begin{aligned} &\text{minimize} \quad c^T x \\ &\text{subject to} \\ &\qquad Ax \le b \\ &\qquad A_{eq} x = b_{eq} \\ &\qquad x \ge 0 \end{aligned}$$

LP

$$\begin{aligned} &\text{minimize} \quad c^T x + d^T y \\ &\text{subject to} \\ &Ax + Gy \le b \\ &A_{eq} x + G_{eq} y = b_{eq} \\ &x \ge 0 \\ &y \ge 0\,(\text{integers}) \end{aligned}$$

MILP

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to} \\ &g(x) \le 0 \\ &h(x) = 0 \\ &lb \le x \le ub \end{aligned}$$

NLP

$$\begin{aligned} &\text{minimize} \quad f(x, y) \\ &\text{subject to} \\ &g(x, y) \le 0 \\ &h(x, y) = 0 \\ &lb \le x \le ub \\ &lb \le y \le ub\,(\text{integers}) \end{aligned}$$

MINLP

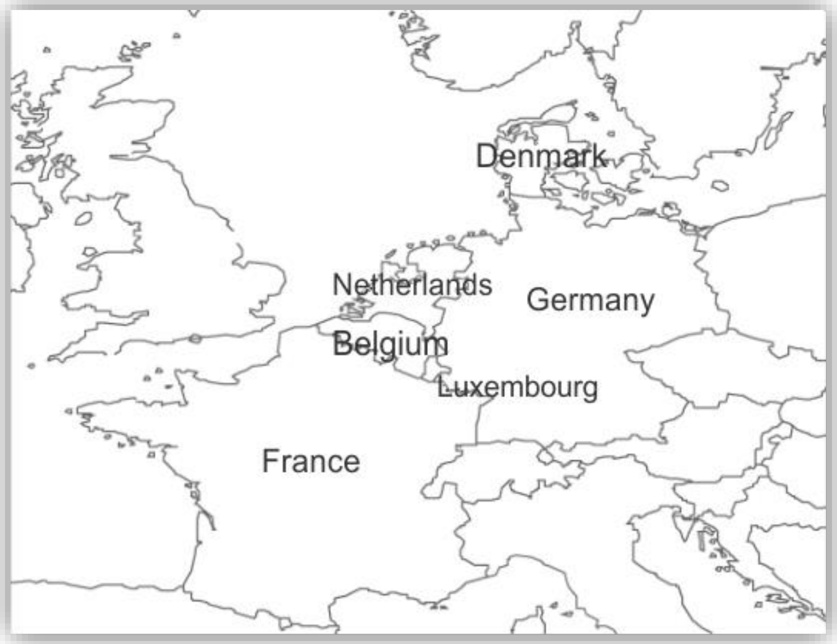
Constraint Programming Applications in IBM ILOG CPLEX Optimization Studio

Map Coloring

- ➢Four sets of colours (blue, white, yellow, green) are used to colour 6 countries such that no pair of neighbouring countries have the same colour

- ➢The object is to find a solution for a map colouring problem with 6 countries: Belgium, Denmark, France, Germany, Luxembourg, and Netherlands

- ➢Neighbours:
  - Belgium: France, Germany, Luxembourg, Netherlands
  - Denmark: Germany
  - France: Belgium, Germany, Luxembourg
  - Germany: Belgium, Denmark, France, Netherlands, Luxembourg
  - Luxembourg: Belgium, France, Germany
  - Netherlands: Belgium, Germany



IBM Knowledge Centre; https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.cpo.help/CP_Optimizer/Getting_started/topics/color_describe.html

CP model

```
using CP;
```

Declaration of engine

```
range r = 0..3;

string Names[r] = ["blue", "white", "yellow", "green"];
```

Declaration of array

```
dvar int Belgium in r;
dvar int Denmark in r;
dvar int France in r;
dvar int Germany  in r;
dvar int Luxembourg in r;
dvar int Netherlands in r;
```

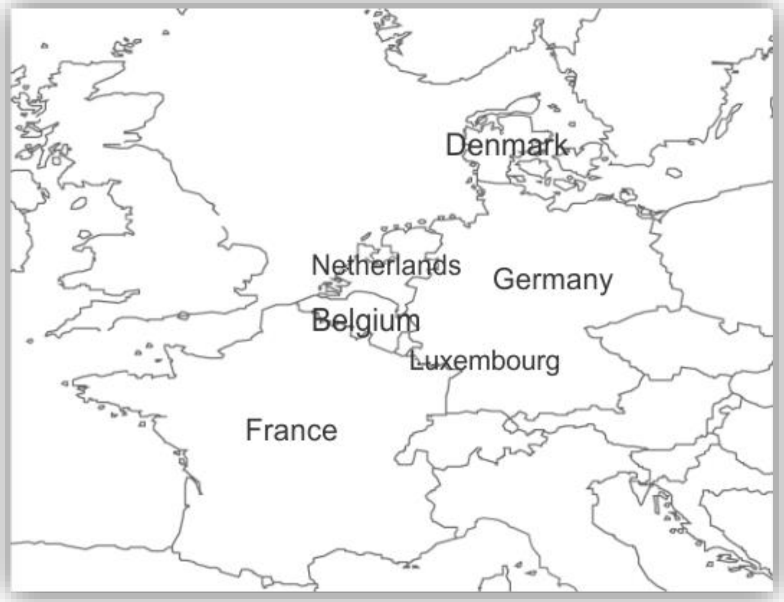
Declaration of decision variables

```
subject to {
  Belgium != France;
  Belgium != Germany;
  Belgium != Netherlands;
  Belgium != Luxembourg;
  Denmark != Germany;
  France != Germany;
  France != Luxembourg;
  Germany != Luxembourg;
  Germany != Netherlands; }
```

Adding the constraints

Belgium = 1;
France = 0;
Germany = 2;
Netherlands = 0;
Luxembourg = 3;
Denmark = 0;

Solution log



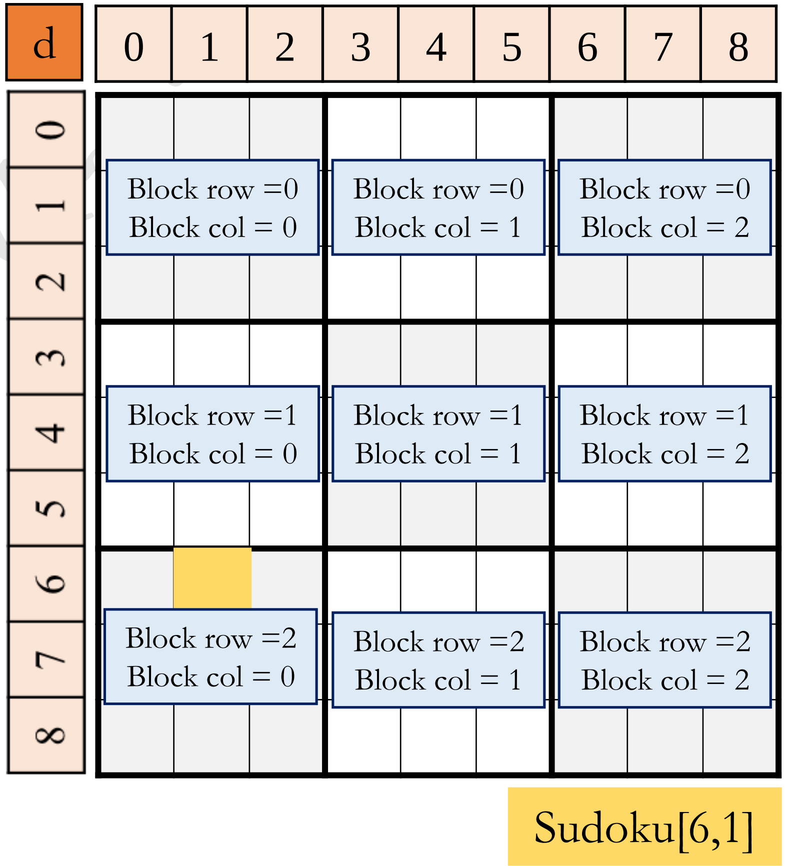
Sudoku: Data file

	7				1	9		
9				6			5	
6	4	1						
	2			7			9	6
		6				7		3
	5			2				
								9
1						6		
			4	3	6			

https://www.ibm.com/developerworks/community/blogs/jfp/entry/solving_the_hardest_sudoku?lang=en

Sudoku: CP model

```
using CP;
range d = 0..8;
int input[d, d] = ...;
dvar int sudoku[d, d] in 1..9 ;
constraints {
   forall (i in d) allDifferent (all (j in d) sudoku[i,j]);
   forall (j in d) allDifferent (all(i in d) sudoku [i,j]);

   forall ( block_row , block_column in 0..2)
      allDifferent (all(l, m in 0..2) sudoku[3 * block_row + l , 3 * block_column + m]);

   forall (i, j in d: input[i, j] != 0) sudoku[i, j] == input[i, j];
};
```



Modelling and solving a house building problem

- **Problem Statement:** Ten tasks need to be completed; Some tasks must necessarily take place before others, and these requirements are expressed through precedence constraints
- **Objective:** No objective function
- **Decision variable:** Determine the start time of each task

Tasks	Duration	Preceding tasks
masonry	35	-
carpentry	15	masonry
plumbing	40	masonry
ceiling	15	masonry
roofing	5	carpentry
painting	10	ceiling
windows	5	roofing
facade	10	roofing, plumbing
garden	5	roofing, plumbing
moving	5	windows, façade, garden, painting

IBM ILOG CPLEX Optimization Studio Getting Started with Scheduling in CPLEX Studio *V12.6* IBM Software

CP model

```
using CP;      Declaration of engine

dvar interval masonry  size 35;
dvar interval carpentry size 15;
dvar interval plumbing size 40;
dvar interval ceiling size 15;
dvar interval roofing size 5;
dvar interval painting size 10;
dvar interval windows size 5;
dvar interval facade size 10;
dvar interval garden size 5;
dvar interval moving size 5;
```

Declaration of data

```
subject to {
  endBeforeStart ( masonry,   carpentry);
  endBeforeStart ( masonry,   plumbing);
  endBeforeStart ( masonry,   ceiling);
  endBeforeStart ( carpentry, roofing);
  endBeforeStart ( ceiling,   painting);
  endBeforeStart ( roofing,   windows);
  endBeforeStart ( roofing,   facade);
  endBeforeStart ( plumbing,  facade);
  endBeforeStart ( roofing,   garden);
  endBeforeStart ( plumbing,  garden);
  endBeforeStart ( windows,   moving);
  endBeforeStart ( facade,    moving);
  endBeforeStart ( garden,    moving);
  endBeforeStart ( painting,  moving);
}
```

Adding the constraints

Result analysis

masonry = <0 35 35>;
carpentry = <35 50 15>;
plumbing = <35 75 40>;
ceiling = <35 50 15>;
roofing = <50 55 5>;
painting = <50 60 10>;
windows = <55 60 5>;
facade = <75 85 10>;
garden = <75 80 5>;
moving = <85 90 5>;

Solution log

Task	Starting time
Masonry	0
carpentry	35
plumbing	35
ceiling	35
roofing	50
painting	50
windows	55
facade	75
garden	75
moving	85

Tasks	Duration	Preceding tasks
masonry	35	-
carpentry	15	masonry
plumbing	40	masonry
ceiling	15	masonry
roofing	5	carpentry
painting	10	ceiling
windows	5	roofing
facade	10	roofing, plumbing
garden	5	roofing, plumbing
moving	5	windows, façade, garden, painting

Thank You !!!