Challenge Decembre – 2020

Virtual Chess Tournament

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)

Let us remember that with ILOG (French company bought by IBM in 2009) we had two kind of decision engines:

A) Rule based (JRules, ODM)
B) Optimization based (CPLEX)

Here a small example of a tiny optimization model, in English, OPL and Python

Zoo, bus, kids and optimization

With words	In OPL	In Python / DoCplex

300 kids need to travel to the London zoo
The school may rent 40 seats and 30 seats buses for 500 and 400 £
How many buses of each to minimize cost ?
 
	int nbKids=300;
	float costBus40=500;
	float costBus30=400;
	
	dvar int+ nbBus40;
	dvar int+ nbBus30;
	
	minimize
	costBus40*nbBus40
	+nbBus30*costBus30;
	
	subject to
	{
	40*nbBus40+
	nbBus30*30
	>=nbKids;
	}
		from docplex.mp.model import Model
		
		mdl = Model(name='buses')
		
		nbbus40 =
		mdl.integer_var(name='nbBus40')
		nbbus30 =
		mdl.integer_var(name='nbBus30')
		
		mdl.add_constraint(nbbus40*40 +
		nbbus30*30 >= 300, 'kids')
		
		mdl.minimize(nbbus40*500 +
		nbbus30*400)
		
		mdl.solve()
		
		print(nbbus40.solution_value);
		print(nbbus30.solution_value);

We can call CPLEX from many languages (C,C++,.NET,Java,Python …) but using OPL leads to a clear frontier between the model and the code that will embed the model. (Not far from Decision Model and Notation (DMN) principle: “The notation is designed to be readable by business and IT users alike. This enables various groups to effectively collaborate in defining a decision model”)

Now let’s move to the Decembre 2020 DMC challenge:

Three world champions Fischer, Kasparov, and Karpov played in a virtual chess tournament. Each player played 7 games against two other opponents. Each player received 2 points for a victory, 1 for a draw, and 0 for a loss. We know that Kasparov, known as the most aggressive player, won the most games. Karpov, known as the best defensive player, lost the least games. And Fischer, of course, won the tournament.

In OPL CPLEX no need to be very clever, we need to translate the constraints.

Let us use a .mod for the model and a .dat to store previous solutions if we find any.

.mod

```
using CP;

// Three world champions Fischer, Kasparov, and Karpov played in a virtual chess
tournament.
{string} players={"Kasparov","Karpov","Fischer"};

tuple game
{
  string player1;
  string player2;
}

{game} games={<i,j> | ordered i,j in players};

tuple previousSolution
{
  int score[players];
}

{previousSolution} previousSolutions=...;

// Each player played 7 games against two other opponents.
range ranks=1..7;

// Each player received 2 points for a victory, 1 for a draw, and 0 for a loss.
dvar int result[games][ranks] in 0..2;
```

```
dvar int nbWins[p in players];
dvar int nbLosses[p in players];
dvar int nbDraws[p in players];
dvar int score[p in players];

subject to
{

forall(p in players)
  {
    nbWins[p]==
    count(all(g in games,r in ranks:g.player1==p)result[g][r],2)+
    count(all(g in games,r in ranks:g.player2==p)result[g][r],0);

    nbLosses[p]==
    count(all(g in games,r in ranks:g.player1==p)result[g][r],0)+
    count(all(g in games,r in ranks:g.player2==p)result[g][r],2);

    nbDraws[p]==
    count(all(g in games,r in ranks:g.player1==p)result[g][r],1)+
    count(all(g in games,r in ranks:g.player2==p)result[g][r],1);

    score[p]==nbWins[p]*2+nbDraws[p];
  }


// We know that Kasparov, known as the most aggressive player, won the most games.
  nbWins["Kasparov"]>nbWins["Karpov"];
  nbWins["Kasparov"]>nbWins["Fischer"];

// Karpov, known as the best defensive player, lost the least games.
  nbLosses["Karpov"]<nbLosses["Fischer"];
  nbLosses["Karpov"]<nbLosses["Kasparov"];

// And Fischer, of course, won the tournament.
  score["Fischer"]>score["Karpov"];
  score["Fischer"]>score["Kasparov"];

  forall(s in previousSolutions)
    or(p in players) s.score[p]!=score[p];
}

previousSolution temp;
execute
{
  writeln("Score = ",score);


  for(var p in players )temp.score[p]=score[p];
  previousSolutions.add(temp);

  writeln("all solutions =",previousSolutions);

  // Write solutions to a file
  var o=new IloOplOutputFile("dmc2020december.dat");
  o.writeln("previousSolutions=");
  o.writeln(previousSolutions);
  o.writeln(";");
```

```
  o.close();
}
```

```
.dat

previousSolutions=
 {}
;
```

```
Which gives {<[14 13 15]> <[13 14 15]>}

So we have 2 solutions and no more.

Within CPLEX we have 2 tools:

Linear programming and constraint programming. Here we used Constraint
Programming.
```

Making Decision Optimization Simple : https://www.linkedin.com/pulse/making-decision-optimization-simple-alex-fleischer/