Challenge April - 2020

Doctor Planning

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

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 Doctor Planning challenge. (April 2020 DMC challenge)

Since we're in the COV19 era, let me thank all doctors and nurses all over the world for what they do.

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

```
range days=1..7;
range weekend=5..7;
int Friday=5;
int Saturday=6;
int Sunday=7;

{string} doctors={"Fleming","Freud","Heimlich","Eustachi","Golgi"};
{string} shifts={"early","late","night"};

assert card(doctors)==5;

tuple t
{
  string doctor;
  int day;
  string shift;
}

{t} availabilities with doctor in doctors =
{<"Fleming",d,s> | d in {Friday,Saturday,Sunday},s in shifts}
union
{<"Freud",d,s> | d in days, s in {"early","late"}}
union
{<"Heimlich",d,s> | d in days,s in shifts : !((d in weekend) && (s=="night"))}
union
{<"Eustachi",d,s> | d in days,s in shifts}
union
{<"Golgi",d,s> | d in days,s in shifts}
;

// is that doctor working that day that shift ?
dvar boolean x[doctors][days][shifts];


// number of shifts per doctor
dvar int nbShifts[doctors];


// minimize max-min of nbShifts
minimize
max(d in doctors) nbShifts[d]-min(d in doctors) nbShifts[d];

// constraints
subject to
{
  // nb of shifts
  forall(d in doctors)
  nbShifts[d]==sum(i in days,s in shifts) x[d][i][s];

  // a doctor can only work one shift a day
```

```
  forall(d in doctors,i in days) sum(s in shifts) x[d][i][s]<=1;

  // specific constraints per doctor

  forall(d in doctors,i in days,s in shifts:<d,i,s> not in availabilities)
    x[d][i][s]==0;

   // max 2 night shifts for Golgi

   sum(d in days) x["Golgi"][d]["night"]<=2;

  // night shift ==> next day off or next night shift

  forall(d in doctors,i in days:(i+1) in days)
   {
     (x[d][i]["night"]==1) => (x[d][i+1]["early"]==0);
     (x[d][i]["night"]==1) => (x[d][i+1]["late"]==0);
  }

  // periodic timetable add on

  forall(d in doctors)
   {
     (x[d][Sunday]["night"]==1) => (x[d][1]["early"]==0);
     (x[d][Sunday]["night"]==1) => (x[d][1]["late"]==0);
  }

  // both days of the week end or none

  forall(d in doctors)
     sum(s in shifts) x[d][Saturday][s]==sum(s in shifts) x[d][Sunday][s];

  //  1 and only 1 doctor per shift

  forall(i in days,s in shifts) sum(d in doctors)  x[d][i][s]==1;

}


string whichDoctor[d in days][s in shifts]=
   first({doc | doc in doctors:x[doc][d][s]==1});

execute display
{
for(var d in days)
{
  write("Day ",d," : ");
  for(var s in shifts) write(whichDoctor[d][s]," ");
  writeln();
}
}
```

Which gives

```
Day 1 : Golgi Freud Heimlich
Day 2 : Freud Eustachi Golgi
Day 3 : Freud Eustachi Heimlich
```

Day 4 : Freud Eustachi Golgi
Day 5 : Heimlich Eustachi Fleming
Day 6 : Golgi Heimlich Fleming
Day 7 : Golgi Heimlich Fleming

NB: April 20th, after an interesting comment from Damir Sudarevic I added a constraint: if you work on Sunday night then you won't work early Monday or late Monday. (Which makes the timetable periodic)

I also added an objective: spread the workload in order to be as fair as possible.

The change took 10 minutes which shows again why relying on an Algebraic Modeling Language helps. (Little model inertia, agility)

https://www.linkedin.com/pulse/optimization-aka-prescriptive-analytics-should-we-write-fleischer/

PS:

Here we relied on Linear Programming. If we want to use Constraint Programming we simply need to add "using CP;" at the beginning of the model

PPS:

This OPL CPLEX model can run on a machine with the free CPLEX community edition but also in the cloud with IBM Watson Machine Learning.

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