Challenge June – 2015

Make a good burger

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)

Optimization can help any kind of business, which includes for sure making burgers.

With OPL CPLEX we can write:

```
tuple foodItem
{
  key string name;
  float sodium;
  float fat;
  float calories;
  float price;
}

{foodItem} foodItems=
{
  <"Beef Patty"           ,50,   17,    220,   0.25>,
  <"Bun"                  ,330,  9,     260,   0.15>,
  <"Cheese",       310,   6,     70,           0.10>,
  <"Onions"               ,1,           2,     10,          0.09>,
  <"Pickles"       ,260,  0,     5,            0.03>,
  <"Lettuce"       ,3,            0,     4,           0.04>,
  <"Ketchup"       ,160,  0,     20,           0.02>,
  <"Tomato"               ,3,           0,     9,           0.04>
};

//at least one of each item and no more than five of each item.
dvar int qty[foodItems] in 1..5;

dvar float cost;
dvar int+ id;

minimize cost;


subject to
```

```
{
// cost
cost==sum(f in foodItems) qty[f]*f.price;
// id allows a 1to1 relation bewteen a number and a solution
id==sum(f in foodItems) (qty[f]-1)*pow(5,ord(foodItems,f));
//  The final burger must contain less than 3000 mg of sodium,

sum(f in foodItems) qty[f]*f.sodium<=3000;

//  less than 150 grams of fat,

sum(f in foodItems) qty[f]*f.fat<=150;

//  and less than 3000 calories.

sum(f in foodItems) qty[f]*f.calories<=3000;

//  To maintain certain taste quality standards you'll need
//  to keep the servings of ketchup and lettuce the same.

qty[<"Ketchup">]==qty[<"Lettuce">];

//  Also, you'll need to keep the servings of pickles and tomatoes the same.

qty[<"Pickles">]==qty[<"Tomato">];



}

execute
{
  writeln("cost = ",cost);
  for(var f in foodItems) writeln(qty[f], " ",f.name);
  writeln();
}

main
{
  thisOplModel.generate();
  // minimize cost

  writeln("minimize cost");
  cplex.solve();
  thisOplModel.postProcess();

  // maximize cost

  writeln("maximize cost");
  thisOplModel.getObjective().setCoef(thisOplModel.cost,-1);
  cplex.solve();
  thisOplModel.postProcess();

  // find all options

  writeln("all possible burgers");

  thisOplModel.getObjective().setCoef(thisOplModel.cost,0);
  thisOplModel.getObjective().setCoef(thisOplModel.id,-1);
```

```
  cplex.epgap=0;
  thisOplModel.id.UB=10000000;
  var nbsol=0;
  while (cplex.solve())
  {
    if (nbsol<=5) thisOplModel.postProcess();
    nbsol++;
    thisOplModel.id.UB=thisOplModel.id.solutionValue-1;
  }


  writeln("and many more");
  writeln("nbsol = ",nbsol);
}
```

Which gives

```
minimize cost
cost = 0.72
1 Beef Patty
1 Bun
1 Cheese
1 Onions
1 Pickles
1 Lettuce
1 Ketchup
1 Tomato

maximize cost
cost = 2.8
5 Beef Patty
5 Bun
1 Cheese
5 Onions
1 Pickles
3 Lettuce
3 Ketchup
1 Tomato

all possible burgers
cost = 2.35
4 Beef Patty
1 Bun
1 Cheese
5 Onions
5 Pickles
5 Lettuce
5 Ketchup
5 Tomato

cost = 2.1
3 Beef Patty
1 Bun
1 Cheese
5 Onions
5 Pickles
```

```
5 Lettuce
5 Ketchup
5 Tomato

cost = 1.85
2 Beef Patty
1 Bun
1 Cheese
5 Onions
5 Pickles
5 Lettuce
5 Ketchup
5 Tomato

cost = 1.6
1 Beef Patty
1 Bun
1 Cheese
5 Onions
5 Pickles
5 Lettuce
5 Ketchup
5 Tomato

cost = 2.26
4 Beef Patty
1 Bun
1 Cheese
4 Onions
5 Pickles
5 Lettuce
5 Ketchup
5 Tomato

cost = 2.01
3 Beef Patty
1 Bun
1 Cheese
4 Onions
5 Pickles
5 Lettuce
5 Ketchup
5 Tomato

and many more
nbsol = 5202
```