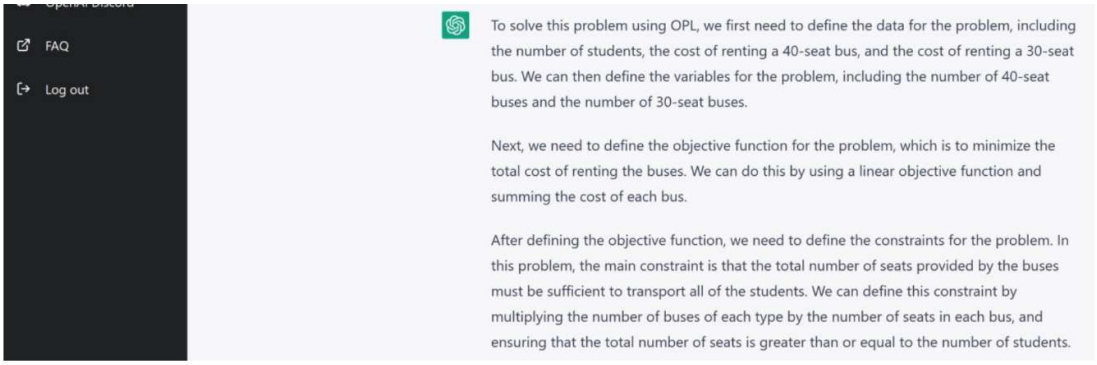
Challenge January 2024
Family Riddle

A solution with chatGPT and docplex CPLEX by Alex Fleischer
afleischer@fr.ibm.com

I am relying on free ChatGPT 3.5 and my first attempt was to use ChatGPT to generate an OPL CPLEX model since in 2022, I managed to use ChatGPT to generate a nice OPL CPLEX model as can be read at https://www.linkedin.com/pulse/optimization-zoo-openai-chatgpt-alex-fleischer/



Optimization, zoo and OpenAI chatgpt

Alex Fleischer
Data and AI Technical Sales

48 articles

December 8, 2022

SC **You**

Let's assume that Rene and Leo are both heads of household, and, what a coincidence, both families include three girls and three boys. The youngest child in Leo's family is a girl, and in Rene's family, a little girl has just arrived. In other words, there is a girl in Rene's family whose age is less than one year. Neither family includes any twins, nor any children closer in age than a year. All the children are under age ten. In each family, the sum of the ages of the girls is equal to the sum of the ages of the boys; in fact, the sum of the squares of the ages of the girls is equal to the sum of the squares of the ages of the boys. The sum of the ages of all these children is 60.

Question: What are the ages of the children in these two families? Write an OPL CPLEX model for that

But the generated OPL CPLEX model is not good enough so that I could improve it. (Garbage)

```
int numChildren = 6;
range Children = 1..numChildren;
int ages[Children] = [1, 2, 3, 4, 5, 6];

dvar boolean isGirl[Children];

minimize sum(i in Children) (ages[i] * ages[i]);

subject to {
  // Constraints related to the number of girls and boys in each family
  count(isGirl, 1) == 3; // Three girls in each family
  count(isGirl, 0) == 3; // Three boys in each family

  // Constraints related to the youngest child
  ages[1] == 1; // Youngest child in Leo's family is a girl
  ages[numChildren] == 1; // Youngest child in Rene's family is a girl

  // Constraints related to the sum of ages
  sum(i in Children) (isGirl[i] * ages[i]) == sum(i in Children) ((1 - isGirl[i]) * ages[i]); // Sum of ages of girls equals sum of ages of boys
  sum(i in Children) ages[i] == 60; // Sum of ages of all children is 60
}
```

```
execute {
  writeln("Optimal solution:");
  writeln("Ages of children in Leo's family:");
  for (i in Children) {
    if (isGirl[i]) {
      writeln("Girl ", i, ": ", ages[i]);
    }
  }

  writeln("Ages of children in Rene's family:");
  for (i in Children) {
    if (!isGirl[i]) {
      writeln("Boy ", i, ": ", ages[i]);
    }
  }
}
```

So my second attempt was to ask for some python code relying on cplex cpoptimizer docplex.

You

Let's assume that Rene and Leo are both heads of household, and, what a coincidence, both families include three girls and three boys. The youngest child in Leo's family is a girl, and in Rene's family, a little girl has just arrived. In other words, there is a girl in Rene's family whose age is less than one year. Neither family includes any twins, nor any children closer in age than a year. All the children are under age ten. In each family, the sum of the ages of the girls is equal to the sum of the ages of the boys; in fact, the sum of the squares of the ages of the girls is equal to the sum of the squares of the ages of the boys. The sum of the ages of all these children is 60.
CAn you write this in docplex cpoptimizer ?

ChatGPT

Certainly! You can use IBM ILOG CPLEX Optimization Studio with the CP Optimizer module to model and solve this problem. Below is an example of how you can write this in Docplex CP Optimizer using Python:

And we get a python model that is a good starting point.

I added

#Break sym

```
model.add(A[0]<A[1])
model.add(A[1]<A[2])

model.add(B[0]<B[1])
model.add(B[1]<B[2])

model.add(C[0]<C[1])
model.add(C[1]<C[2])

model.add(D[0]<D[1])
model.add(D[1]<D[2])
```

in order to break symmetries

Then I fixed the all different

```
# Constraint: No twins or children closer in age than a year
model.add(model.all_diff(A[0],A[1],A[2],B[0],B[1],B[2]))
model.add(model.all_diff(C[0],C[1],C[2],D[0],D[1],D[2]))
```

Not to forget

```
# Constraint: Sum of squares of ages is equal for girls and boys in
each family
model.add(model.sum([A[i] * A[i] for i in range(3)]) ==
model.sum([B[i] * B[i] for i in range(3)]))

model.add(model.sum([C[i] * C[i] for i in range(3)]) ==
model.sum([D[i] * D[i] for i in range(3)]))
```

which was nearly ok but wrong.

And finally I added

#And then to enumerate all solutions we write

```
siter = model.start_search(SearchType='DepthFirst', Workers=1,
TimeLimit=100,log_output=None)
# Parameters needed to avoid duplicate solutions

nbsol=0
for msol in siter:
    nbsol=nbsol+1
    print("solution ",nbsol)
    print("Ages in Rene's family:", [msol[A[i]] for i in
range(3)],[msol[B[i]] for i in range(3)])
    print("Ages in Leo's family:", [msol[C[i]] for i in
range(3)],[msol[D[i]] for i in range(3)])
```

which gives

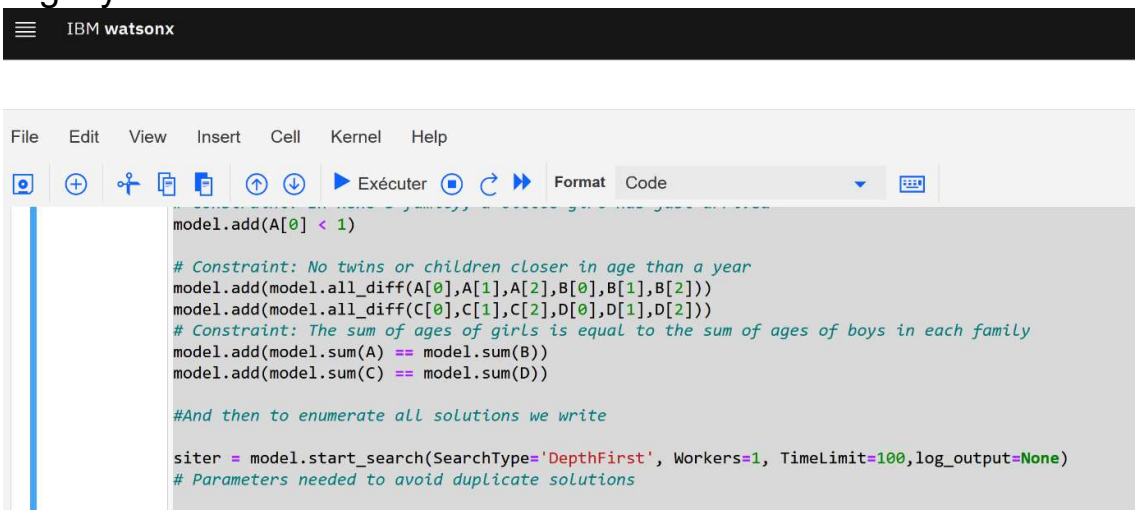
```
from docplex.cp.model import CpoModel

# Create a CPO model
model = CpoModel()

# Define variables representing the ages of the children
A = model.integer_var_list(3, 0, 9, name='A')
B = model.integer_var_list(3, 0, 9, name='B')
C = model.integer_var_list(3, 0, 9, name='C')
D = model.integer_var_list(3, 0, 9, name='D')

# Constraint: Sum of ages in each family is 60
model.add(model.sum(A + B + C + D) == 60)

# Constraint: Sum of squares of ages is equal for girls and boys in
each family
model.add(model.sum([A[i] * A[i] for i in range(3)]) ==
model.sum([B[i] * B[i] for i in range(3)]))

model.add(model.sum([C[i] * C[i] for i in range(3)]) ==
model.sum([D[i] * D[i] for i in range(3)]))
```

```
#Break sym

model.add(A[0]<A[1])
model.add(A[1]<A[2])

model.add(B[0]<B[1])
model.add(B[1]<B[2])

model.add(C[0]<C[1])
model.add(C[1]<C[2])

model.add(D[0]<D[1])
model.add(D[1]<D[2])

# Constraint: The youngest child in Leo's family is a girl
model.add(C[0] < D[0])

# Constraint: In Rene's family, a little girl has just arrived
model.add(A[0] < 1)

# Constraint: No twins or children closer in age than a year
model.add(model.all_diff(A[0],A[1],A[2],B[0],B[1],B[2]))
model.add(model.all_diff(C[0],C[1],C[2],D[0],D[1],D[2]))
# Constraint: The sum of ages of girls is equal to the sum of ages of
boys in each family
model.add(model.sum(A) == model.sum(B))
model.add(model.sum(C) == model.sum(D))

#And then to enumerate all solutions we write

siter = model.start_search(SearchType='DepthFirst', Workers=1,
TimeLimit=100,log_output=None)
# Parameters needed to avoid duplicate solutions

nbsol=0
for msol in siter:
    nbsol=nbsol+1
    print("solution ",nbsol)
    print("Ages in Rene's family:", [msol[A[i]] for i in
range(3)],[msol[B[i]] for i in range(3)])
```

```
    print("Ages in Leo's family:", [msol[C[i]] for i in
range(3)],[msol[D[i]] for i in range(3)])
```

Which I can run within IBM watsonx or any python IDE

In green, what I had to write from scratch and in red what I had to slightly fix.



And I see

```
solution  1
Ages in Rene's family: [0, 5, 7] [1, 3, 8]
Ages in Leo's family: [3, 7, 8] [4, 5, 9]
```

In a nutshell, OPL CPLEX and ChatGPT was a dead end.
OPL CPLEX alone took me 30 minutes.
ChatGPT + python docplex took me one hour.