# Wal-timise Challenge

August 8, 2020

### 0.0.1 Team Name: Black_Bishop

### 0.0.2 Members:


- Rakesh Pavan, 7th sem, B.Tech Information Technology (email:- rp.171it154@nitk.edu.in, ph:- 8277539885)
- Dhruvik Navadiya, 7th sem, B.Tech Information Technology


### 0.0.3 Method Used: Mixed-Integer Programming

1. We first formulate the problem as a mixed-integer optimization problem.
2. The objective function being to minimize the total hit value:

$$\min \sum_{i=1}^{300} H_i^T X_i$$

where X is the required assignment / solution, and H is the hit value corresponding to each assignment, both in suitable encoding.

3. Subject to the constraints being on revenue percentage, and quantity percentage.

$$\frac{\sum_{i=0}^{300} R_i^T X_i - \text{Base\_Revenue}}{\text{Base\_Revenue}} \geq x$$

$$\frac{\sum_{i=0}^{300} Q_i^T X_i - \text{Base\_Quantity}}{\text{Base\_Quantity}} \geq y$$

where R and Q capture the revenue, and quantity associated with each item, again everything in suitable encoding.

4. Our choice of was multi-staged. First we converted to one-hot encoding. Then we flatten the resulting 2-D vector into 1-D by laying it out linearly. Resulting in additional 300 constraints to make sure each item is assigned to only 1 price (out of the 5 possible options).

5. Thus, the formulated optimization problem is on 1500 (= 300 x 5) binary-variables (vector X), subject to 302 constraints.

6. We implement and solve this mixed-integer program using the Coin-or-branch and Cut (CBC), which is an open-source mixed integer programming solver in C++.
7. The OR-Tools is a library by Google, providing an interface to several third-party mixed-integer programming (MIP) solvers. The Coin-or branch and cut (CBC) solver is included in OR-Tools.
8. Our solution takes only **13-14 seconds** to solve this problem (tested on Core i7 8th Gen laptop, with 16 GB RAM). Thus, is it very efficient and can be scaled to much larger problem sizes.

### 0.0.4 Code:

[ ]:
```
from __future__ import print_function
from ortools.linear_solver import pywraplp

import numpy as np
import pandas as pd



'''
    Scenario 1. X = 10%, Y = 25%
    For other scenarios change the below x,y,scen variables
'''

x = 0.10
y = 0.25
scen = 'scenario1'

print('X={0}%, Y={1}%, Scenario = {2} \n'.format(np.round(x*100),np.
 ↪round(y*100),scen))






# Reading and processing the data
data = pd.read_csv('dataset.csv')
data = data.drop(columns=['Item_id'])
data = data.to_numpy()
data = data.T

R = []
Q = []
H = []

for i in range(5):
  R.append(data[2*i]*data[2*i+1])
```

```
  Q.append(data[2*i+1])
  H.append(np.abs(data[2*i]-data[0])*data[2*i+1])

R = np.array(R)
Q = np.array(Q)
H = np.array(H)

base_revenue = np.sum(R[0])
base_quantity = np.sum(Q[0])

R = R.T
Q = Q.T
H = H.T

for i in range(len(R)):
  R[i] = R[i]-R[i][0]
  Q[i] = Q[i]-Q[i][0]

reven = R.flatten()
quan = Q.flatten()
hit = H.flatten()


more_cons = np.zeros((300,1500))


for i in range(300):
  for j in range(5):
    more_cons[i][5*i+j] = 1


more_bnds = np.ones(300)

bounds = []
cons = []

cons.append(reven)
bounds.append(x*base_revenue)
cons.append(quan)
bounds.append(y*base_quantity)

for i in range(300):
  cons.append(more_cons[i])
  bounds.append(more_bnds[i])

tmp = []
```

```python
def mip_formulation():
  """Preparing the MIP formulation"""
  data = {}
  data['constraint_coeffs'] = cons
  data['bounds'] = bounds
  data['obj_coeffs'] = hit
  data['num_vars'] = 1500
  data['num_constraints'] = 302
  return data


def main():
  data = mip_formulation()
  # Create the mip solver with the CBC backend.
  solver = pywraplp.Solver.CreateSolver('simple_mip_program', 'CBC')
  infinity = solver.infinity()

  # Setting the variables
  x = {}
  for j in range(data['num_vars']):
    x[j] = solver.IntVar(0, 1, 'x[%i]' % j)
  print('Number of variables =', solver.NumVariables())

  # Adding the constraints
  for i in range(data['num_constraints']):
   constraint_expr = [data['constraint_coeffs'][i][j] * x[j] for j in range(data['num_vars'])]
   if i<2:
    solver.Add(sum(constraint_expr) >= data['bounds'][i])
   else:
    solver.Add(sum(constraint_expr) == data['bounds'][i])
  print('Number of constraints =', solver.NumConstraints())

  # Defining the objective
  obj_expr = [data['obj_coeffs'][j] * x[j] for j in range(data['num_vars'])]
  solver.Minimize(solver.Sum(obj_expr))

  # Calling the solver
  status = solver.Solve()

  if status == pywraplp.Solver.OPTIMAL:
    print('Final Hit value obtained=', solver.Objective().Value())
    for j in range(data['num_vars']):
      # print(x[j].name(), ' = ', x[j].solution_value())
      tmp.append(x[j].solution_value())
    print()
    print('Problem solved in %f milliseconds' % solver.wall_time())
```

```python
  else:
    print('The problem does not have an optimal solution.')


if __name__ == '__main__':
  main()

  # Processing the output
  tmp = np.array(tmp)
  sols = []
  for i in range(300):
    for j in range(5):
      if tmp[5*i+j] > 0.5:
        sols.append(j)
        break

  lst = range(1,301)

  # Saving the solution
  df = pd.DataFrame(list(zip(lst,sols)),columns = ['Item_id','Price'])
  df['Price'] = df['Price'].map({0:'Base_Price',1:'Price1',2:'Price2',3:
↪'Price3',4:'Price4'})
  df.to_csv(scen + '.csv',index=False)
```