ORIE 51350 - Project - Professor Andrea Lodi

Negar Akbarzadeh (na332)

May 5, 2022

We are given a set of n machines (j = 1, . . . , n) and a set of m jobs (i = 1, . . . ,m). Each job is characterized by a processing time $p_i \in R^+$ and by a memory requirement $r_i \in R^+$. The processing time and memory requirement of a job do not depend on the machine in which the job is executed. All machines are identical and characterized by a time capacity P and a memory capacity R. We assume that $p_i \leq P$ and $r_i \leq R$ for all i = 1, . . . ,m.
We want to execute all jobs on the machines in such a way that the number of used machines is minimized.

Questions:

1- Propose a "natural" (descriptive) Integer Linear Programming (ILP) formulation and implement it.
Answer:

Data:
A: Set of machines (A = $\{j|j = 1, .., n\}$)
B: Set of jobs (B = $\{i|i = 1, .., m\}$)
p_i = processing time for job i to completely run
r_i = required memory for job i to be done
P = total time capacity
R = total memory capacity
$p_i \leq P$ and $r_i \leq R$ $\forall$ i = 1, . . . ,m

Varibales:

$$X_{ij} = \begin{cases} 1 & \text{if job i is done by machine j} \\ 0 & \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{if machine j is used} \\ 0 & \text{otherwise} \end{cases}$$

Model:
min $\sum_{j \in B} y_j$
s.t.

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad \text{i=1,..,m}$$

$$X_{ij} \leq y_j \qquad \forall i \in A, j \in B$$

$$\sum_{i=1}^{m} p_i X_{ij} \leq P y_j \qquad \text{j=1,..,n}$$

$$\sum_{i=1}^{m} r_i X_{ij} \leq R y_j \qquad \text{j=1,..,n}$$

$$X \in \{0,1\}^{mn} \qquad y \in \{0,1\}^{n}$$

This model has many constraints. It also can be written in this format which has less linear constraints where M is sufficiently a large constant which makes the RHS to the largest possible amount i.e. if $y_j = 1$, the RHS is $M.1 = M$

min $\sum_{j\in B} y_j$

s.t.

$\sum_{j=1}^{n} X_{ij} = 1$ i=1,..,m

$\sum_{i=1}^{m} X_{ij} \leq My_j$ $\forall j \in B$

$\sum_{i=1}^{m} p_i X_i j \leq Py_j$ j=1,..,n

$\sum_{i=1}^{m} r_i X_{ij} \leq Ry_j$ j=1,..,n

X$\in \{0,1\}^{mn}$ y$\in \{0,1\}^{n}$

2- Propose an extended formulation of the problem with a variable per feasible job assignments to machines. Implementing such a formulation as well, assuming to be able to enumerate all the feasible job assignments to machines.

Answer:
$\mathbb{S}$:= {all feasible solutions} If S is a set of all feasible solutions where all jobs can be run by minimum amount of machines. Therefore, X_s can be defined as the following:

$$X_s^j = \begin{cases} 1 & \text{if the job s} \in \mathbb{S} \text{ is covered by machine j} \\ 0 & \text{otherwise} \end{cases}$$

min $\sum_{s\in\mathbb{S}} X_s^j$

s.t.

$\sum_{j\in S} X_s^j \geq 1$

$X_s^j \in \{0,1\}^{|S|}$

3- Solve the ILP formulations of questions 1 and 2 above on the three provided datasets (10 instances each). Collect and report data on
• average quality of the LP relaxations,
• average number of branch-and-bound nodes, and
• average computing times and number of problems solved in either 120 CPU seconds or 300 CPU seconds.
Based on the above data, compare the two formulations.
Answer:
To analyze the difference between the performance of these two models, we need to take a look at the details of performance of each model in each dataset.
Model of first question has many constraints. In the first dataset which P=150, R=150, n=15 and m=15 , we have 240 binary variables and 270 constraints. In the second dataset we have P=150, R=150 , n= 30 and m=15 where the number of variables is 930 (binary) and the of constrains is 990. In the third dataset, which P=150, R=150, n=50 and m=50 , we have 2550 binary variables and 2650 constraints. This is obvious that the feasible solution for this problem is that only one machine covers all jobs.
In details,the objective bounds for the instance 0 in dataset 1 is reduced to 7.14% after 115 times iterations which shows there is a gap between the lower bound (6.5) and the upper bound.
The details of this comparison are as follows in the following table.

instance number	Optimal objective value of LP without relaxation	Optimal objective value of LP with relaxation	run time(s)
0	7	5.50820	0.0171571
1	6	6	0.00931001
2	6	6	0.0113602
3	6	4.91143	0.0161121
4	8	6.27860	0.00945616
5	6	4.67913	0.0100231s
6	4	2.99175	0.010875
7	7	7	0.0561268
8	6	4.95838	0.0305569
9	5	3.82601	0.010344
		Average run time :	0.018132137s

For dataset 2:

instance number	Optimal objective value of LP without relaxation	Optimal objective value of LP with relaxation	run time(s)
0	12	11.79659	0.0202041
1	12	11.53069	0.02033
2	14	13.79995	0.0176489
3	15	14.25197	0.0193391
4	11	11	0.0240672
5	13	12.5	0.0203161
6	14	14	0.020716
7	10	10	0.0238771
8	13	13	0.0220561
9	10	10	0.02649
		Average run time :	0.02150446s

For dataset 3:

instance number	Optimal objective value of LP without relaxation	Optimal objective value of LP with relaxation	run time(s)
0	10	8	0.102319
1	10	9	0.092572
2	10	9.33	0.0842149
3	10	9.66	0.087049
4	11	9.66	0.0726831
5	10	9.66	0.080641
6	9	8	0.0856199
7	9	7	0.0994639
8	10	9.66	0.110919
9	9	8	0.0841322
		Average run time :	0.0899614s

1- The quality of LP relaxation:
The quality of relation in the first model is increased by 15% in dataset 1,17% in dataset 2 and 10% in dataset3.
2- The average number of branch and bound nodes is $\frac{1}{10} = 0.1$.
3- The average computing times:
The average model runtime based on the gurobi attribute in $model.printAttr('runtime')$is shown the the tables.
All problems (30 instances) are solved in less than 120 and 300 CPU seconds in the first model. The second model have better performance in general than the first model.

4- Assume now that instead of minimizing the number of used machines we want to minimize the completion time of the machine, among those used, that finishes the latest. The completion time for a machine j is given by the sum of the processing times of the jobs executed by machine j. For this variant of the problem,

(a) Write a "natural" (descriptive) ILP formulation and implement it.

Answer:

If we had a single machine, we would have this formulation:

min $\sum_{j\in B} x_j$

s.t.

$$x_i + p_j - x_j \leq M z_{ij} \qquad \forall i \in A, j \in B, i \leq j$$

$$x_j + p_i - x_i \leq M(1 - z_{ij}) \qquad \forall i \in A, j \in B, i \leq j$$

$$x_j \geq p_j \qquad \forall j \in B$$

$$z_{ij} \in \{0,1\} \qquad \forall i \in A, j \in B, i < j$$

$$x_j \geq 0 \qquad \forall j \in B$$

where M is a upper bound such as $\sum_{j\in B} p_j$.

In General with several machines and jobs, we will have this formulation:

Let's define x_{ij} binary vaiable as follows:

$$x_{ij} = \begin{cases} 1 & \text{if job i is done by machine j} \\ 0 & \text{otherwise} \end{cases}$$

And define the completion time of machine j as follows:

On machine $j \in B$, the last job is completed at $\sum_{i\in A} p_i x_{ij}$

The maximum completion time among all machines defines as follows :

$w \geq \sum_{i\in A} p_i x_{ij}$

Therefore, the formulation for this problem can be written in this form:

min w

s.t.

$$w \geq \sum_{i\in A} p_i x_{ij} \qquad \forall j \in B$$

$$\sum_{j\in B} x_{ij} = 1 \qquad \forall i \in A$$

$$\text{x}_{ij} \in \{0,1\} \qquad \forall i \in A, j \in B$$

(b) Solve the ILP formulation on the provided single dataset (10 instances), collect and report data as for point 3 above.

Answer : Unfourtunatley, I couldn't run the code for my formulation but the code with details shows as follows:

```
    %matplotlib inline
!pip install gurobipy

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sys
import gurobipy as gp
from gurobipy import GRB
```

```python
from time import time

def read_data():
    data = pd.read_csv('constants.csv')
    R = data['value'][0]
    n = data['value'][1]
    return (R,n)

def instance_data(number_of_instance):
    instance = pd.read_csv('instance_'+str(number_of_instance)+'.csv')
    m = instance.shape[0]
    p = []
    r = []
    for p_i in instance['p_i']:
        p.append(p_i)
    for r_i in instance['r_i']:
        r.append(r_i)
    return (m,p,r)

def constructVars_q4():
    global n , m , model_q4, xVars , w
    for i in range( m ):
        for j in range( n ):
            var = model_q4.addVar( vtype = GRB.BINARY , name = "x_" + str( i+1 ) + "_" + str( j+1 ) )
            xVars[ i , j ] = var
    model_q4.update()

    for j in range( n ):
        var_2 = model_q4.addVar( vtype = GRB.BINARY , name = "w_" + str( j+1 ))
        w[ j ] = var_2
    model_q4.update()

def constructObj_q4():
    global n , m , model_q4, xVars , w
    objExpr = gp.LinExpr()
    for j in range( n ):
        objExpr += w
    model_q4.setObjective( objExpr , GRB.MINIMIZE )
    model_q4.update()

def contructConstrs_q4():
    global n , m , model_q4, xVars , R, w
    for i in range( m ):
        constExpr_0 = gp.LinExpr()
        for j in range( n ):
            constExpr_0 += 1.0 * xVars[ i,j ]
        model_q4.addConstr( lhs = constExpr_0 , sense = GRB.EQUAL , rhs = 1 )

    for j in range( n ):
        constExpr_2 = gp.LinExpr()
        rhs_exp_2 = gp.LinExpr()
        for i in range( m ):
            constExpr_2 += p[j] * xVars[ i,j ]
            rhs_exp_2 += p[j]
        model_q4.addLConstr( lhs = constExpr_2 , sense = GRB.GREATER_EQUAL , rhs = rhs_exp_2  )
    model_q4.update()

def Model_q4( number_of_instance , m , p , r , R , n ):
    global model_q4, xVars ,w
    model_q4 = gp.Model("q1_instance_"+str(number_of_instance))
    xVars = {}
    w = {}
    for i in range(n):
        for j in range(m):
            xVars[i,j] = 0
    for i in range(m):
        w [i] = 0
    constructVars_q4()
    contructConstrs_q4()
    constructObj_q4()
    model_q4.write('test.lp')
    model_q4.optimize()
    model_q4.printAttr('X')
    model_q4.printAttr('Xn')
    model_q4.printAttr('ObjVal')
    model_q4.printAttr('Status')

number_of_instance = 0
m , p , r = instance_data( number_of_instance )
R,n = read_data()
print('\n instnce '+str(number_of_instance)+'\n--------------------')
Model_q4( number_of_instance , m , p , r , R , n)
print('\n----------------------------------------------------------------\n\n')
```