Operations Research Documentation

Release 1.0

Qiuyi Hong

Oct 30, 2021

LINEAR ALGEBRA

CHAPTER

ONE

INTRODUCTION

Contents

1.1 Sub section

1.1.1 Sub subsection

$$k_{\mathrm{eq}} = (2\Omega_{\mathrm{M}} H_0^2 z_{\mathrm{eq}})^{1/2}$$

This is math equation $\sum_{n=1}^{N} x^2 + y^2 = 1$

$$\frac{x}{y}$$

```
for i in range(10):
    print(i)
```

Note: This is a note!

CHAPTER

TWO

VECTORS

Contents

2.1 Scalars

2.2 Vectors: geometry and algebra

2.3 Transpose Operation

2.4 Vector addition and subtraction

2.5 Vector-scalar multiplication

Go and check *Sub subsection*

Check doc *Introduction*

CHAPTER

THREE

VECTOR MULTIPLICATION

Contents

3.1 Vector dot product: Algebra

CHAPTER

FOUR

OPTIMIZATION ALGORITHMS

Contents

4.1 The Simplex Method

4.2 The Branch and Bound Algorithm

4.3 Gradient Descent and Newton's Method

4.4 Examples

4.4.1 Machine Scheduling Problem

Fifteen jobs, each with its processing time, should be scheduled on three machines. If two jobs cannot be scheduled on the same machine, they are called conflicting jobs. Table 1 lists the job IDs, processing times, and sets of conflicting jobs. For example, we cannot schedule any pair of jobs out of jobs 2, 5, 8 on the same machine. Note that a job may have no conflicting jobs.

Table 1: Table 1: Data

Job	Processing time	Conflicting jobs
1	7	None
2	4	5,8
3	6	None
4	9	None
5	12	2,8
6	8	9
7	10	10
8	11	2,5
9	8	6
10	7	7
11	6	15
12	8	None
13	15	None
14	14	None
15	3	11

We want to schedule the jobs to minimize makespan. For example, we may schedule jobs 1, 4, 7, 8, and 13 to machine 1, jobs 2, 6, 10, 11, and 14 to machine 2, and jobs 3, 5, 9, 12, and 15 to machine 3. The total processing times on the three machines are 52, 39, and 37, respectively. The makespan is thus 52. While this is a feasible schedule, this may or may not be an optimal schedule. When we try to improve the schedule, be careful about conflicting jobs. For example, we cannot exchange jobs 8 and 11 (even though this reduces the makespan) because that will result in machine 2 processing conflicting jobs 2 and 8, which is infeasible.

Formulate a linear integer program that generates a feasible schedule to minimize makespan. Then write a computer program (e.g., using Python to invoke Gurobi Optimizer) to solve this instance and obtain an optimal schedule. Write down the minimized makespan (i.e. the objective value of an optimal solution).

$$\min_{X_{i,j},w} w$$

subject to:

$$w \geq \sum_{j \in \mathcal{J}} t_j X_{i,j}, \forall i \in \mathcal{I}$$

$$\sum_{i \in \mathcal{I}} X_{i,j} = 1, \forall j \in \mathcal{J}$$

$$\sum_{j \in \Omega_{1,2,3,4}} X_{i,j} \leq 1, \forall i \in \mathcal{I}$$

where $i \in \mathcal{I}$ denotes machine index. $j \in \mathcal{J}$ denotes job index.

Conflicting job sets are $\begin{cases} \Omega_1 & \{2,5,8\} \\ \Omega_2 & \{6,9\} \\ \Omega_3 & \{7,10\} \\ \Omega_4 & \{11,15\} \end{cases}$

Decision variable $X_{i,j} = \begin{cases} 1 & \text{if job } j \text{ is allocated to machine } i \\ 0 & \text{otherwise} \end{cases}$

Decision variable w is a trivial positive value.

```
import numpy as np
import cvxpy as cp
```

(continues on next page)

(continued from previous page)

```
X = cp.Variable(shape=(3,15),boolean=True)
w = cp.Variable()

t = np.array([[7,4,6,9,12,8,10,11,8,7,6,8,15,14,3]]).T

obj = cp.Minimize(w)

constrs = []

constrs += [w >= X[0,:]@t]
constrs += [w>= X[1,:]@t]
constrs += [w >= X[2,:]@t]

for j in range(15):
    constrs += [X[0,j]+X[1,j]+X[2,j] == 1]

for i in range(3):
    constrs += [X[i,1]+X[i,4]+X[i,7] == 1]
    constrs += [X[i,5]+X[i,8] <= 1]
    constrs += [X[i,6]+X[i,9] <= 1]
    constrs += [X[i,10]+X[i,14] <= 1]

prob = cp.Problem(obj, constrs)

prob.solve(solver=cp.GUROBI)
```

4.4.2 Facility Location Problem

A city is divided into n districts. The time (in minutes) it takes an ambulance to travel from District i to District j is denoted as d_{ij}. The population of District i (in thousands) is p_i. An example is shown in Table 2 and Table 3. The distances between districts are shown in Table 2, and the population information is shown in Table 3. In this instance, we have $n = 8$ districts. We may see that, e.g., it takes 5 minutes to travel from District 2 to District 3, and there are 40,000 citizens.

Table 2: Table 2: distances

District	1	2	3	4	5	6	7	8
1	0	3	4	6	8	9	8	10
2	3	0	5	4	8	6	12	9
3	4	5	0	2	2	3	5	7
4	6	4	2	0	3	2	5	4
5	8	8	2	3	0	2	2	4
6	9	6	3	2	2	0	3	2
7	8	12	5	5	2	3	0	2
8	10	9	7	4	4	2	2	0

Table 3: Table 3: District Population

District	Population
1	40
2	30
3	35
4	20
5	15
6	50
7	45
8	60

The city has mm ambulances and wants to locate them to mm of the districts. For each district, the population-weighted firefighting time is defined as the product of the district population times the amount of time it takes for the closest ambulance to travel to it. The decision maker aims to locate the mm ambulances to minimize the maximum population-weighted firefighting time among all districts.

As an example, suppose that $m = 2$, $n = 8$, $d_{i,j}$ and p_i are provided in Table 2, and the two ambulances are located in District 1 and 8. We then know that for Districts 1, 2, and 3 the closest ambulance is in District 1 and for the remaining five districts the closet ambulance is in District 8. The firefighting time for the eight districts are thus 0, 3, 4, 4, 4, 2, 2, and 0 minutes, respectively. The population-weighted firefighting times may then be calculated as 0, 90, 140, 80, 60, 100, 90, and 0. The maximum among the eight districts is therefore 140.

For this problem, formulate an integer program that can minimize the maximum population-weighted firefighting time among all districts. Then write a program to invoke a solver (e.g., write a Python program to invoke Gurobi Optimizer) to solve the above instance and find an optimal solution for each problem. Write down the minimized maximum population-weighted firefighting times among all districts of the two districts that ambulances should be located in (i.e., the objective value of an optimal solution).

$$\min_{X_{i,j},w} w$$

subject to:

$$\sum_{j \in \mathcal{N}} x_j = m$$
$$Y_{i,j} \leq x_j, \forall i, j \in \mathcal{N}$$
$$\sum_{j \in \mathcal{N}} Y_{i,j} = 1, \forall i \in \mathcal{N}$$
$$w \geq \sum_{j \in \mathcal{N}} d_{i,j} p_i Y_{i,j}, \forall i \in \mathcal{N}$$
$$x_i, Y_{i,j} \in \{0, 1\}, \forall i, j \in \mathcal{N}$$
$$w \geq 0$$

where Decision variable $Y_{i,j} = \begin{cases} 1 & \text{if for District } i \text{ the cloest ambulance is located in District } j \\ 0 & \text{otherwise} \end{cases}$

Decision variable $x_j = \begin{cases} 1 & \text{if an ambulance is located in District } j \\ 0 & \text{otherwise} \end{cases}$

Decision variable w is a trivial positive value.

```
import numpy as np
import cvxpy as cp
```

(continues on next page)

(continued from previous page)

```
m = 2
n = 8

d = np.array([[0,3,4,6,8,9,8,10],
              [3,0,5,4,8,6,12,9],
              [4,5,0,2,2,3,5,7],
              [6,4,2,0,3,2,5,4],
              [8,8,2,3,0,2,2,4],
              [9,6,3,2,2,0,3,2],
              [8,12,5,5,2,3,0,2],
              [10,9,7,4,4,2,2,0]])

p = np.array([[40,30,35,20,15,50,45,60]]).T

x = cp.Variable((n,1), boolean=True)
Y = cp.Variable((n,n), boolean=True)
w = cp.Variable()

obj = cp.Minimize(w)

constrs = [cp.sum(x) == m]

for i in range(n):
    constrs += [cp.reshape(Y[i,:],(1,n)) <= x.T]

for i in range(n):
    constrs += [cp.sum(Y[i,:]) == 1]

for i in range(n):
    constrs += [w >= d[i,j]*p[i]*Y[i,j] for j in range(n)]


prob = cp.Problem(obj, constrs)

prob.solve(solver=cp.GUROBI)
```

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search