Report of project 2

尚尔淦 (PB19010390)

Contents

1 introduction

We first give the **description** of the problem in reality:

We now have a excavation task which is assigned for several machines(equipment) of number m to several sites of number n. Moves between different places and the set-off of every machines will have additional cost.

In this problem, we assume that one machine needs to do the task in at least one site and every site is needed to be done only once. And in the process of moving a machine, this machine cannot disturb other machines working in other sites. Now we display the optimization problem as follows:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} \\ s.t. \quad & \sum_{i\in V_s^+} x_{si} = m \\ & \sum_{j\in V_i^-} x_{ji} = \sum_{k\in V_i^+} x_{ik} = 1, \forall i \in V\backslash\{s,t\} \\ & x_{ij} \in \{0,1\}, \forall (i,j) \in E \end{aligned}$$

where V is the set of nodes and E are the edges; the first constrain means that from the source there are m vehicles(equipment) setting out; the second constraint means that except the source and meeting node, the in-flowing and out-flowing of nodes are equal.

2 Input and output

2.1 Input

- Cost Matrix: which is between the vehicles(equipment) and the places(tasks)($C_1 \in \mathbb{R}^{m\times n}$)(named as cep in codes)

equipment	place1	place2	place3	place4	place5
X001	272	116	105	156	227
X002	213	232	270	263	232
X003	113	283	180	142	276

- Cost Matrix: which is among places(tasks)($C_2 \in \mathbb{R}^{n\times n}$)(named as cpp in codes)

equipment	place1	place2	place3	place4	place5
Place1	0	126	227	256	299
Place2	275	0	100	128	233
Place3	169	249	0	263	109
Place4	223	242	200	0	177
Place5	173	206	265	191	0

2.2 Output

- Assignment sheme and result of the code:
  - Assignment graph

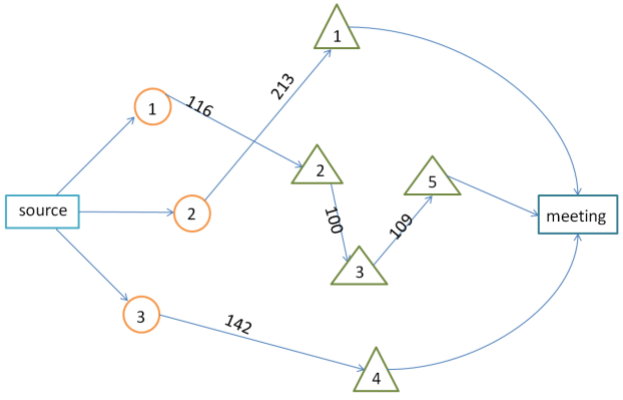

Figure 1: Assignment Graph

- Results from code: Now we display the results from my code(using **ortools**) to verify the graph above is really true(for details please read my code attached in the zip)

```
%time main(cep,cpp)
```

```
vehicle 1 go to place 2 with cost 116
vechile 1 next go to place 3 with cost 100
vechile 1 next go to place 5 with cost 109
route of vehicle 1 cost 325
vehicle 2 go to place 1 with cost 213
route of vehicle 2 cost 213
vehicle 3 go to place 4 with cost 142
route of vehicle 3 cost 142
total cost= 680.0
Wall time: 12 ms
```

Figure 2: outputs from codes

From the code-result displayed above, we can get the following sheme list:

- X001 $\overset{116}{\rightarrow}$ place2 $\overset{100}{\rightarrow}$ place3 $\overset{109}{\rightarrow}$ place5: with the cost 325
- X002 $\overset{213}{\rightarrow}$ place1: with the cost 213
- X003 $\overset{142}{\rightarrow}$ place4: with the cost 142

- Totla Cost: From the code above, the total cost is **680**.
- Time consumed: The result is 12ms by using **%time**

3 How to run the code

We just copy the input matrix in a list of python, where cep means the cost between equipment andplaces; cpp means the cost between places.

```
cep=[[272,116,105,156,227],
     [213,232,270,263,232],
     [113,283,180,142,276]]#cep means cost from equuipment to place
cpp=[[0,126,227,256,299],
     [275,0,100,128,233],
     [169,249,0,263,109],
     [223,242,200,0,177],
     [173,206,265,191,0]]#cpp means cost from place to place
```

Finally, just using the main function with the parameter of the matrices you input, then you can see the result mentioned in the section 2.2 in this report.

```
%time main(cep,cpp)
```