EX3

Ido Kahana

October 2021

1 Q3 Part 1

Definition 1.1 — Formal model,

- Let $n \in \mathbb{N}$ be the number of taxi's, We have $4n$ people to drive
- Let $P = \{1, \dots, 4n\}$ be the set of all passengers
- Let $T = \{1, \dots, n\}$ be the set if all taxis
- The cost matrix is represented using $C \in \mathbb{R}^{n \times 4n}$ when $C_{i,j} \geq 0$ is the fuel amount it cost for driver i to take passenger j
- The output is represent using $A \in P^{n \times 4}$, where $A_{i,j} \in P$ is the passenger that taxi i should take at it j drive.
- Let L be the Group of all the legal assignments

$$L = \{U \mid U \in P^{n\times4} \wedge (\{U_{i,k} \mid i \in T, k \in \{1,2,3,4\}\} = P)\}$$

Algorithm 1 Passengers assignments

Input: $C \in \mathbb{R}^{n\times4n}$,$P, T$
(P, T can be constructed using size of C no need to really pass them as parameter)

```
 1: Define $S = T \times \{1,2,3,4\}$ (times taxi can take passenger)
 2: Define $\overline{E} = S \times P$ (edges without costs)
 3: Define $m = 1 + \arg\max C_{i,j}$ (Value of the must expensive passenger + 1)
 4: Define $E = \{((i,k), j, m - C_{i,j}) \mid ((i,k), j) \in \overline{E}\}$ (edges with costs)
 5: Define $V = (S \cup P)$ (vertexes with costs definition)
 6: Define $G = (V, E)$ (graph with costs definition)
 7: $R$ = find perfect maximum weighted pair matching for $G$
 8: for $((i,k), j) \in S$ do
 9:     $A_{i,k} = j$
10: end for
11: return $A$
```

Theorem 1.2 — *Passengers assignments algorithm provide legal assignment.*
Formally, Consider some C, P, T, n and A an output from Passengers assignments. Then

$$A \in L = \{U \mid U \in P^{n\times4} \wedge (\{U_{i,k} \mid i \in T, k \in \{1,2,3,4\}\} = P)\}$$

Proof.

- The algorithm vertexes is compose from $V = (S \cup P)$, S represented places of each taxi, P represented passengers, edges are only possible between S and P, any edge have positive utility since

$$\forall_{i \in T, j \in P} m - C_{i,j} > 0 \iff \forall_{i \in T, j \in P} m > C_{i,j} \iff \forall_{i \in T, j \in P} 1 + \arg\max C_{r,s} > C_{i,j}$$

- The size of $|P| = 4n$ and also $|S| = |T \times \{1,2,3,4\}| = 4n$, therefore G is bipartite graph with two equals sides and edges with positive utilities, so it have a maximum weighted perfect matching, therefore such R exists.

- R which is a matching will match a place from S to each passenger from P, for such R there must be an edge to each passenger in P and it is impossible that an vertex will have 2 edges in R, or in other words it is impossible that passenger will be taken twice by the same taxi or by different taxi, formally that is

$$\forall_{((i_1,k_1),j_1) \in R, ((i_2,k_2),j_2) \in R} \quad (i_1 = i_2 \iff j_1 = j_2) \land (i_1 = i_2 \land j_1 = j_2 \implies k_1 = k_2)$$

so

$$\forall_{j \in P} \exists_{i,k} \quad ((i,k),j) \in R \implies \forall_{j \in P} \exists_{i,k} \quad A_{i,k} = j \quad \implies \{A_{i,k} \mid i \in T, k \in \{1,2,3,4\}\} = P$$

□

Theorem 1.3 — *Passengers assignments algorithm is effective, that is it find the minimum assignments in terms of fuel consumption of passngers to taxi's.*
Formally, Consider some C, P, T, n and A an output from Passengers assignments. Then

$$\forall_{B \in L} \sum_{i \in T} \sum_{k \in \{1,2,3,4\}} C_{i,A_{i,k}} \le \sum_{i \in T} \sum_{k \in \{1,2,3,4\}} C_{i,B_{i,k}}$$

Proof. Suppose in the negative that,

$$\exists_{B \in L} \sum_{i \in T} \sum_{k \in \{1,2,3,4\}} C_{i,A_{i,k}} > \sum_{i \in T} \sum_{k \in \{1,2,3,4\}} C_{i,B_{i,k}}$$

Let $B \in L$ (some legal assignment) be such that it upholds the claim above,
let $m = (1 + \arg\max C_{i,j})$, then using the equation above we can conclude that

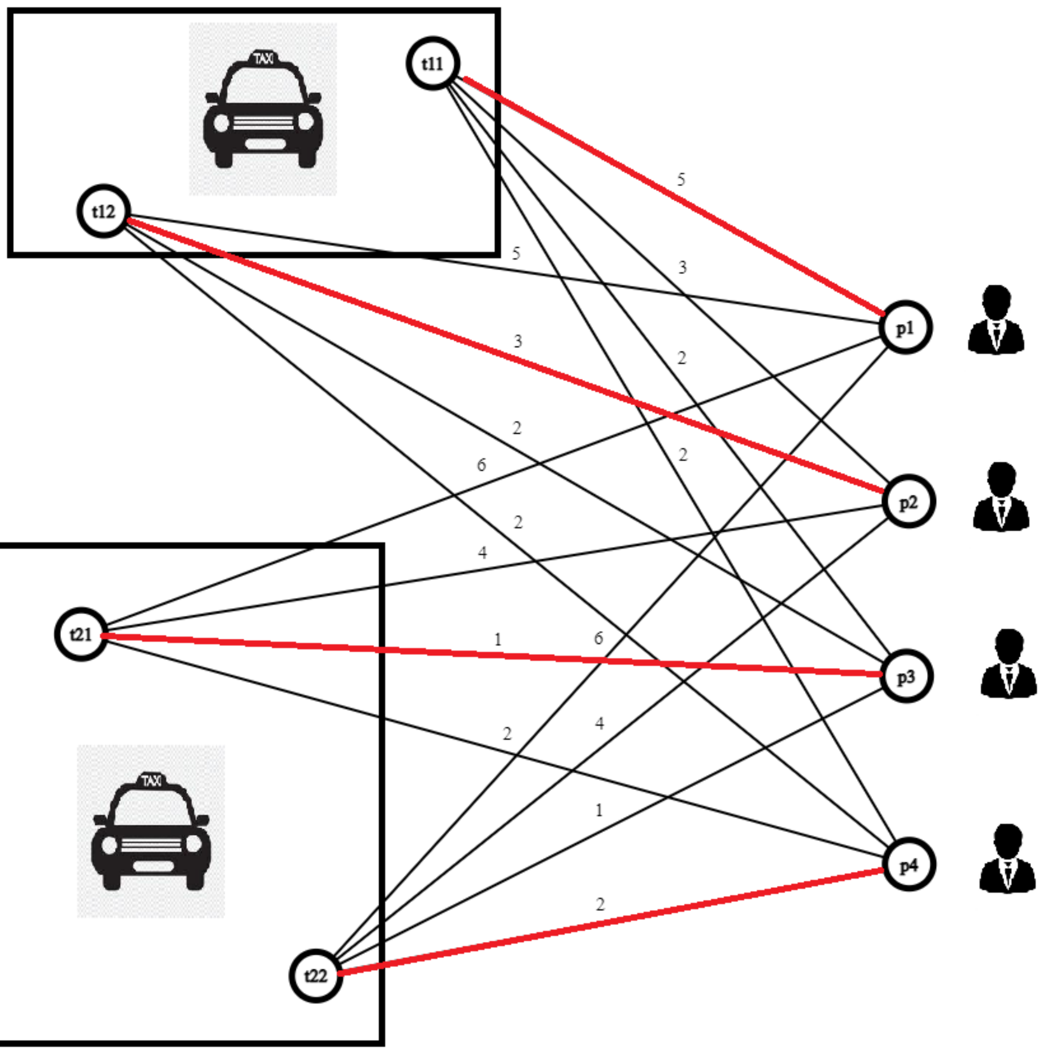
$$\sum_{i \in T} \sum_{k \in \{1,2,3,4\}} C_{i,A_{i,k}} > \sum_{i \in T} \sum_{k \in \{1,2,3,4\}} C_{i,B_{i,k}} \implies \sum_{i \in T} \sum_{k \in \{1,2,3,4\}} -C_{i,A_{i,k}} < \sum_{i \in T} \sum_{k \in \{1,2,3,4\}} -C_{i,B_{i,k}} \implies$$

$$\sum_{i \in T} \sum_{k \in \{1,2,3,4\}} m - C_{i,A_{i,k}} < \sum_{i \in T} \sum_{k \in \{1,2,3,4\}} m - C_{i,B_{i,k}}$$

So we can construct the following matching $\overline{R} = \{((i,k),j) \mid B_{i,k} = j\}$, and it have higher value then the matching from line 7, that is R, and that a contradiction to the fact that R is maximum weighted matching. □

There is an example in the next page, and Part 2 is in the next-next page

2 Q3 Part 1 Example



Example for graph building for case when there are 2 passengers per taxi and the utility matrix is [[5,3,2,2],[6,4,1,2]]

3 Q3 Part 2

```
import networkx as nx
from networkx import max_weight_matching

def find_assigment(C):
    G = nx.Graph()
    max_value = 0

    # negative utilities could be an issue so i find the maximum utility and add a bias
    for i, a in enumerate(C):
        for j, b in enumerate(a):
            max_value = max(max_value, C[i][j])

    # add one so it will be for sure bigger
    max_value += 1

    # building bipartite graph
    for i, a in enumerate(C):
        for j, b in enumerate(a):
            for k in range(4):
                G.add_weighted_edges_from([((i, k), j, max_value - C[i][j])])

    assignments = [[0 for x in range(4)] for y in range(len(C))]
    cost_per_driver = [0 for y in range(len(C))]
    total_cost = 0

    for c in max_weight_matching(G):
        a = c[0]
        b = c[1]

        # some bug in implementation the vars position in the tuple are changing
        if isinstance(b, int):
            i = a[0]
            k = a[1]
            j = b
        else:
            i = b[0]
            k = b[1]
            j = a

        total_cost += C[i][j]
        cost_per_driver[i] += C[i][j]
        assignments[i][k] = j

    print()
    print()
    print("input: ", C)
    print("graph: ", G)
    print("assignments: ", assignments)
    print("total cost: ", total_cost)
    print("cost per driver: ", cost_per_driver)

find_assigment([[1, 2, 3, 4, 5, 60, 7, 8],[4, 2, 3, 1, 8, 7, 6, 5]])
find_assigment([[10, 20, 30, 40, 50, 60, 7000, 80],[4, 2, 3, 1, 8, 7, 6, 5]])
find_assigment([[10, 2, 3, 4, 5, 60, 7, 8, 700, 600, 300, 0],
                [40, 2, 3, 1, 8, 7, 6, 5, 50, 80, 90, 100],
                [40, 2, 3, 1, 8, 7, 6, 5, 11, 60, 90, 5]])
```