ISE 754_Midterm 2

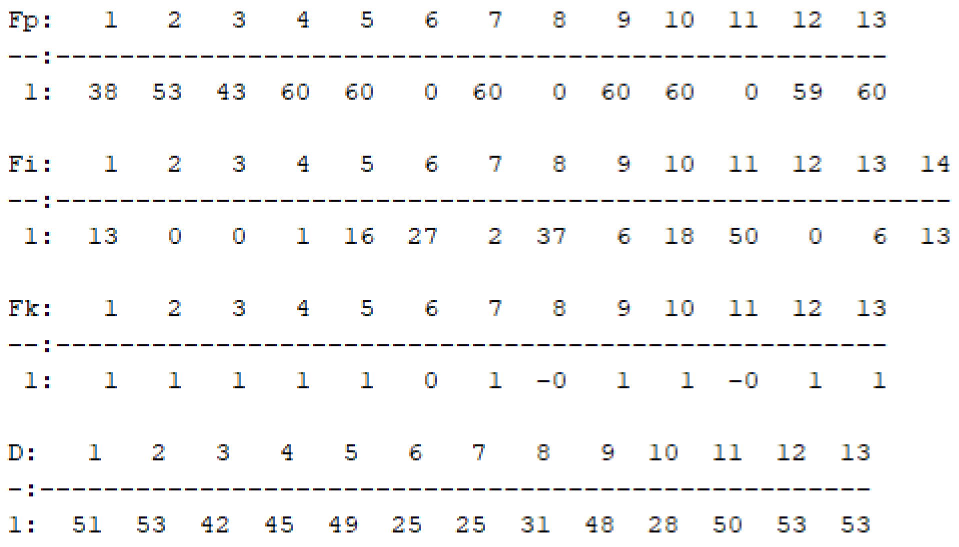
Kimia Vahdat

(#200262784)

Question 1:

a) Overall approach and final results:

This is a network problem with 1 product, 1 stage and 13 weeks period. So, my overall approach is very similar to the Network 3 script. The difference here is that we do not have production cost, instead we have historical data of production and its cost. Therefore, I used this data to estimate a fixed and a variable production cost using linear regression. The final result is summarized in the figure below.



```
Fp:   1    2    3    4    5    6    7    8    9   10   11   12   13
--:-------------------------------------------------------------
 1:  38   53   43   60   60    0   60    0   60   60    0   59   60

Fi:   1    2    3    4    5    6    7    8    9   10   11   12   13   14
--:------------------------------------------------------------------
 1:  13    0    0    1   16   27    2   37    6   18   50    0    6   13

Fk:   1    2    3    4    5    6    7    8    9   10   11   12   13
--:-------------------------------------------------------------
 1:   1    1    1    1    1    0    1   -0    1    1   -0    1    1

D:   1    2    3    4    5    6    7    8    9   10   11   12   13
-:-------------------------------------------------------------
1:  51   53   42   45   49   25   25   31   48   28   50   53   53
```

Where, Fp is production amount in each period, Fi is inventory in each period, Fk is a binary variable indicating periods that we have production in and at last, D is the demand given in each period.

b) Assumptions:

- It is assumed that production fixed costs (k) only occur when we have production in a period and otherwise we do not have any fixed cost.
- To calculate h (inventory carrying rate), I assumed the interest rate is 5% and warehousing rate is 6% per year. h= interest rate + warehousing rate + obsolescence rate. Calculation of the third term is explained in the next part.
- In the linear programming section of the code, since we only have one stage, the flow balance constraints only need to maintain balance between production and inventory of each period and the demand of that period. To be more precise, the constraint we need to include in model is:

$$x_t + y_t - y_{t+1} = D_t$$

- In this linear programming we have 3 variables, x (production amount), y (inventory) and k, which is a binary variable that only takes values if we have production in that period. This can be included into our model by this constraint:
$$x_t \leq kK_t$$
Where, k is the production capacity and K is just a binary variable indicating production.
- To include the inventory capacity in the model, I set an upper bound on the inventory variable (y).

c) Description of Procedure:

First, we need to get the production fixed cost and variable cost, to do so, used the historical data given and fit a regression model to it. The intercept of that model is our fixed cost and the slope is our variable cost. I got 5.8812e+04$ for fixed cost per week and 798.4160$ for cost of production per ton*week. To get inventory carrying rate, we need to calculate each of the terms of h= interest rate + warehousing rate + obsolescence rate. As mentioned before, I assumed interest rate to be 5% and warehousing to be 6% per year. To get obsolescence rate, we know that the product loses 50% its value after 4 weeks, so its rate is: 0.5/4=0.125 per week. Now to get h, all we need to do is to convert their units to be per week (assuming each year has 12*4=48 weeks) and sum them up: h=0.05/12*4+0.06/12*4+0.5/4= 0.1273 per week.

After getting all the data needed, we need to construct our linear programming model. The objective function for this problem is:
$$\min \sum_t Cp * x_t + Ci * y_t + k * K_t$$
Where Cp=798.4160$, k=5.8812e+04$ and Ci=101.6317$. The constraints are those mentioned in the assumption section. Then I solved this with Gorubi and got the result displayed on section "a" of this report.

Question 2:

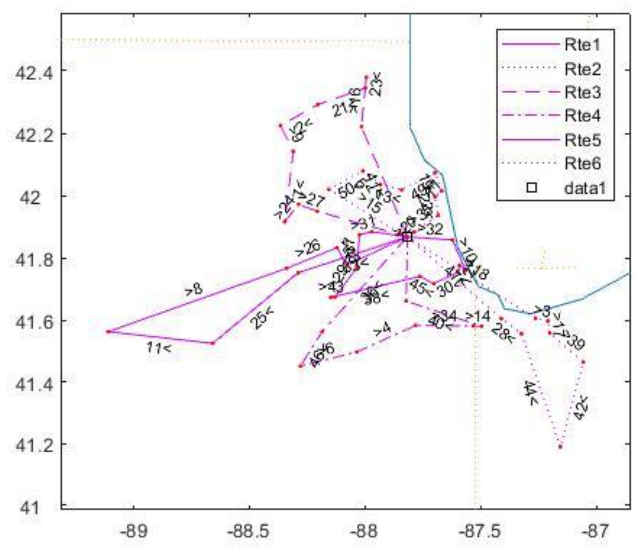
I have 2 approaches in solving this question. My first approach is to ignore the one hour charging time and calculate routes just based on other constraints and at the end combine whichever routes have total time of less than 9 hours including charging time at the hub, in this approach I need 5 vans to do the delivery.

The second approach I used is to make all the routes to be finished in total 4 hours (9 hours total-1 hour charge=8 hour total travel time and divide 8 by 2 so we get 4 hours for each section of route), then all vans must stop at the hub in the middle of the day to recharge and take the next route. At the end we can combine each 2 of these routes together to form a complete route (since we are now sure that it would be feasible time wise). For this approach I got 4 for final number of vans needed.

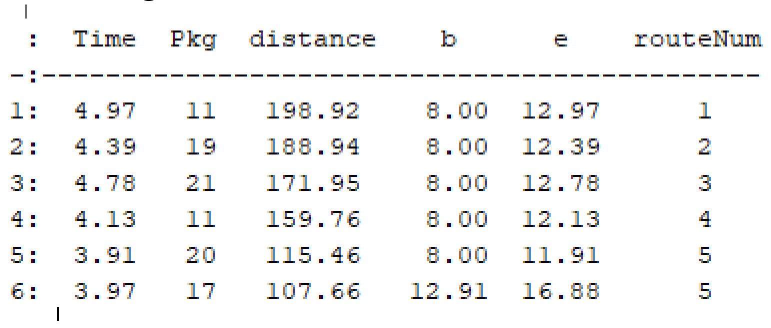
First Approach

a) Overall approach and final results:

This is a routing problem similar to the one we had on the HW 8. Except in this problem we have one more constraint, which is the total distance a van can travel without recharging is 200 miles. Since we have two kinds of constraint in this problem, both time constraint and distance constraint, I wrote my own function of route total cost, which considers both of these constraints. At first, I did not consider the 1 hour charging time and calculated the optimal routes, the program found 6 optimal routes which are shown in the graph below:

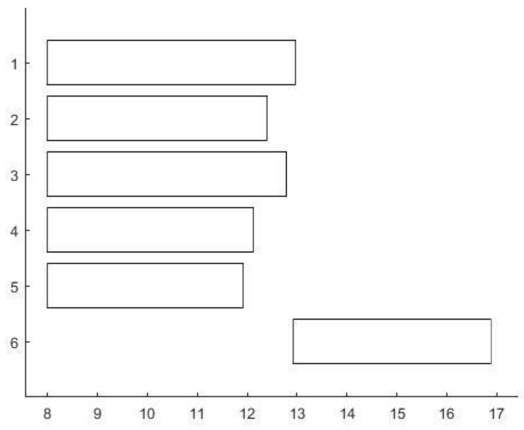


Then, I searched through all the routes created to see which of them have summation of times smaller than 8 (9 hour total time -1 hour charging at the hub). I found out that only route 5 and 6 can be summed together. So, one van can do their routes together and 4 other vans can do the rest of the routes. Results for routes (Time, Distance traveled and packages delivered) are shown in figure below.



```
 :  Time   Pkg  distance    b      e     routeNum
-:--------------------------------------------------
1:  4.97   11   198.92    8.00  12.97       1
2:  4.39   19   188.94    8.00  12.39       2
3:  4.78   21   171.95    8.00  12.78       3
4:  4.13   11   159.76    8.00  12.13       4
5:  3.91   20   115.46    8.00  11.91       5
6:  3.97   17   107.66   12.91  16.88       5
```

We can see that distance traveled is a bounding constraint in almost all of routes. Therefore, we need a total of 5 vans for these routes, and their starting time and ending time is also shown below in the gantt chart.



b) Assumptions:

- First, I assumed the distances between customers and depot can be calculated using road network script given in MakeNetwork 2. And minimum time traveled is also calculated in this script, with default values.
- To include both constraints, to our model, I defined a new rteTC function named "myrteTC.m" (included in the zip file). In this function, I first check the distance function by calling rteTC with distance matrix as its input and then if it satisfies the constraint (D<200) I calculate the travel time using the same function rteTC, but with matrix time as an input. And set TC to be the time traveled.
- I assumed by calculating routes without considering charging, each route will fulfill its maximum traveling distance , and then we can combine routes that are feasible, so each section of a route (going from hub and returning back to hub is considered as one section) will utilizes maximum distance it can travel. I assume this would give us a near optimal solution.

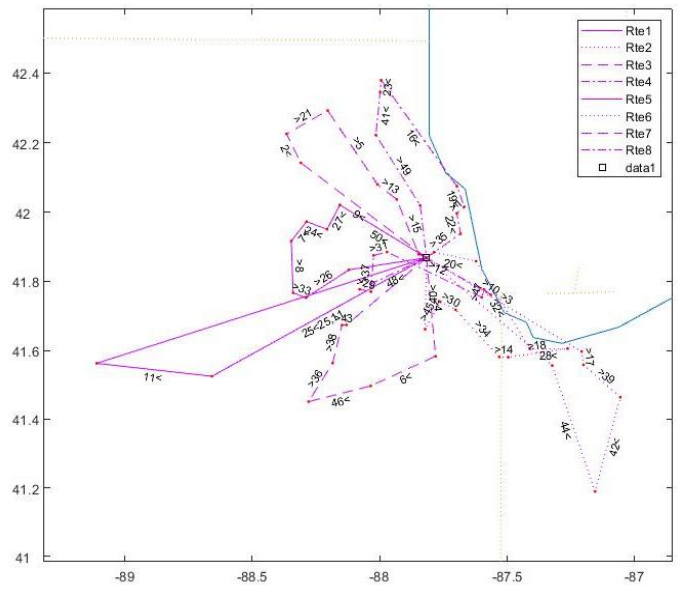
c) Description of Procedure:

First, I combined the hub location to customers' locations to a single matrix and created the road network with default values used in the script. Then, to construct and improve routes, I defined a new function named myrteTC. This function takes the values "sh","tr","D","T","maxdistance" and "route" as inputs and give total cost in terms of time as the output. The way it calculates is, first it calls rteTC function in matlog with D as input cost and compares this distance given by the function to the maximum distance allowed to travel. If the distance is larger than maximum it sets the TC to be infinity and otherwise it sets it to be the output of rteTC with T as its input. For this part of the code I added the maximum total time traveled constraint to the tr structure so it would be considered in calculation. After defining myrteTC, I used the usual process of creating and improving routes to get the optimal routes. After that, I checked if any points are left out of all routes and create a single route for them. At last, I report each routes' time and distance and see if these routes can be combined together, meaning their summation of time is smaller than 8. Only 2 of these routes can be combined together route 5 and 6, so I combine these two together and the final time and schedule is shown in part a.

Second Approach:

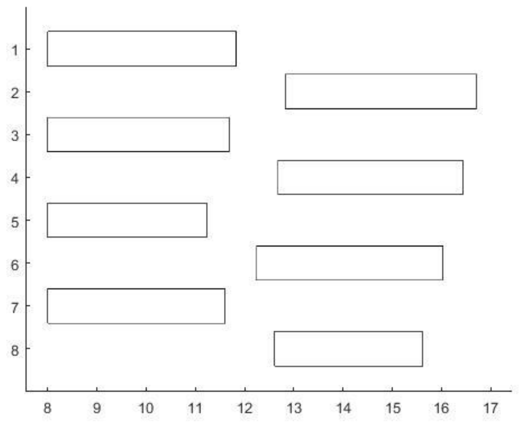
a) Overall approach and final results:

As explained before, in this approach I changed the time window constraint from 9 hours (17pm-8am=9) to 4 hours so I would be able to combine each 2 routes together at the end and get full routes which all of them return to hub in the middle and recharge. After doing this, I followed the rest of the steps the same as first approach. The final routes and their distance and time and packages are shown below.



```
 :   Time   Pkg   distance     b       e     routeNum
-:---------------------------------------------------
1:   3.83     3    199.98    8.00   11.83        1
2:   3.89    13    183.97   12.83   16.72        1
3:   3.68    14    125.99    8.00   11.68        2
4:   3.75    14    132.81   12.68   16.43        2
5:   3.23    15    105.20    8.00   11.23        3
6:   3.79    15    131.66   12.23   16.02        3
7:   3.61    11    139.42    8.00   11.61        4
8:   3.01    14    103.00   12.61   15.62        4
```

I also attached the Gantt chart for this approach to see how we are able to combine these routes. In the chart 1 hour charging time is considered.



Since we have a total 8 routes, after combining each two of them we would get 4 full routes which would need 4 vans.

b) Assumptions:

The assumptions here are the same as before, so I would not repeat them. The only difference is that the total travel time has changed to 4 in “myrteTC2.m” (included in the zip file) so we could have the result shown above.

Instead of last assumption in the previous approach, this time I assumed if we utilize the maximum amount of the time we have on hand (i.e. 9 hours) we have to use less vans, since each van is used for almost 8 hours in each route, and this might give us a near optimal solution.

Depending on the goal of the Green Van Delivery Inc. whether they want minimum number of vans or they want less working hours, they can choose either of these approaches.

c) Description of Procedure:

The procedure is the same, except we are using a different custom function named myrteTC2, in which the “maxTC” in the truck structure is set to 4. Also, at the end of the code, for each 2 routes in a row, I changed the beginning and end time so each couple of them would be assigned to a single van.