Industrial Engineering Department

Final Carreer Project

PHARMACY SUPPLY OPTIMIZATION

Application for Kutahya center, Turkey.

Víctor Aragón Caminero

victoraragoncam@gmail.com

CONTENT TABLE

1. PROJECT OBJECTIVE.

The objective of this project is to find a cheaper and sustainable way to deliver provisions to pharmacies in order that final customer price is reduced, vehicular traffic in the city center is smaller and therefore, pollution owing to transport decreases in the center of Kutahya.

2. CURRENT SUPPLY SYSTEM.

With the help of a native student we talked with the city centre pharmacy managers. They informed us about how the supply system worked:

When a pharmacy needs a product, they contact the depot. If the depot has enough product, they send a motorbike or a van to the pharmacie (according with quantity).

A. Electric motorbike.
   Small orders and normal or low demand products; high frecuency (at least once a day); weight of the order [1 - 3], 40 kg a month in total of averaje. This represents normally less than the 20% of the total weight ordered a month.

B. Van.
   Generally for big orders and high demand products; frecuency varies according with the demand and the pharmacy capacity, (on weekends, twice a week, even daily…); weight of the order is [3 - 5] kg per cage and the average number of cages ordered in a day is [2 - 4].

- In winter the weight increases drastically (almost 3 times more than the average) because of liquid medicies such as syrup for the cough - .

Else, they contact with some wholesaler or some truck on the way, to deliver it directly to the pharmacy.

Observing this data, we notice this is a considerably dinamical system which is difficult to study.

Moreover, we could see that normally there is no collaboration among pharmacies since each one make their own orders independently from the others, except some cases when they are really close (pharmacies 1, 2, 10 and 11).

3. PROPOSED SUPPLY SYSTEM.

For reaching our objetives, we propose a different system in which pharmacies work together renting vehicles for collecting ware from wholesalers, bring it to depot and deliver from depot to pharmacies.

This would be the procedure:

When a pharmacie ask depot for some product, this will look in the aviable stock first. If it has not got, the pharmacy will be able to suggest the depot where to buy it and depot will compare with the cheapest offers in the markets. This order will be public so that every pharmacy is able to join it during some time, specifying the amount needed. When the time expires, the depot sends vehicles to collect (solving "Vehicle Routing Problems) all the products needed from the wholesalers - or another delivery vehicles - and transport them till depot. Once in depot, some vehicles describe again another optimal routes, this time for delivering the required cuantity of ware to each pharmacy.

Each one of the pharmacies should study which is the optimal quantity of ware to buy. This implies to make a statistical analysis for estimating how many stock might need in depot when an emergency occurs. For example, when demand increases in winter and the capacity of the shop is not enough or when they simply run out of a product owed to a peak demand.

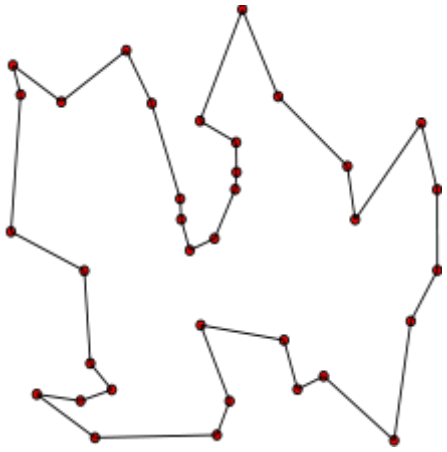
4. THEORETICAL BASIS.

This project focuses on the latest part of this system, the delivery from depot to pharmacies, which consists in solving a "Vehicle Routing Problem". However, even more convenient would be to find the VRP solution for the wholesalers collecting stage due to distances are much longer so savings will be quite greater.

4.1. TRAVELING-SALESMAN PROBLEM – The Root Problem.

4.1.1. Description.

The travelling salesman problem (TSP) asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.



Solution of a travelling salesman problem.

In the theory of computational complexity, the decision version of the TSP (where, given a length L, the task is to decide whether the graph has any tour shorter than L) belongs to the class of NP-complete problems. Thus, it is possible that the worst-case running time for any algorithm for the TSP increases exponentially with the number of cities.

The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, a large number of heuristics and exact methods are known, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%. [1]

4.1.2. Applications.

o Drilling of printed circuit boards.

A direct application of the TSP is in the drilling problem of printed circuit boards (PCBs) (Grötschel et al., 1991).To connect a conductor on one layer with a conductor on another layer, or to position the pins of integrated circuits, holes have to be drilled through the board. The holes may be of different sizes. To drill two holes of different diameters consecutively, the head of the machine has to move to a tool box and change the drilling equipment. This is quite timeconsuming. Thus it is clear that one has to choose some diameter, drill all holes of the same diameter, change the drill, drill the holes of the next diameter, etc. Thus, this drillingproblem can be viewed as a series of TSPs, one for each hole diameter, where the 'cities' are the initial position and the set ofall holes that can be drilled with one and the same drill. The 'distance' between two cities is given by the time it takes to move the drilling head from one position to the other. The aim isto minimize the travel time for the machine head. [2]

o Overhauling gas turbine engines.

(Plante et al., 1987) reported this application and it occurs when gas turbine engines of aircraft have to be overhauled. To guarantee a uniform gas flow through the turbines there are nozzle-guide vane assemblies located at each turbine stage. Such an assembly basically consists of a number of nozzle guide vanes affixed about its circumference. All these vanes have individual characteristics and the correct placement of the vanes can result in substantial benefits (reducing vibration, increasing uniformity of flow, reducing fuelconsumption). The problem of placing the vanes in the best possible way can be modeled as a TSP with a special objective function. [2]

o X-Ray crystallography.

Analysis of the structure of crystals (Bland & Shallcross, 1989; Dreissig & Uebach, 1990) is an important application of the TSP. Here an X-ray diffractometer is used to obtain information about the structure of crystalline material. To this end a detector measures the intensity of X-ray reflections of the crystal from various positions. Whereas the measurement itself can be accomplished quite fast, there is a considerable overhead in positioning time since up to hundreds of thousands positions have to be realized for some experiments. In the two examples that we refer to, the positioning involves moving four motors. The time needed to move from one position to the other can be computed very accurately. The result of the experiment does not depend on the sequencein which the measurements at the various positions are taken. However, the total time needed for the experiment depends on the sequence. Therefore, the problem consists of finding a sequence that minimizes the total positioning time. This leads to a traveling salesman problem. [2]

o Computer wiring.

(Lenstra & Rinnooy Kan, 1974) reported a special case of connecting components on a computer board. Modules are located on a computer board and a given subset of pins has to be connected. In contrast to the usual case where a Steiner tree connection is desired, here the requirement is that no morethan two wires are attached to each pin. Hence we have the problem of finding a shortest Hamiltonian pathwith unspecified starting and terminating points. A similar situation occurs for the so-called testbus wiring. To test the manufactured board one has to realize a connection which entersthe board at some specified point, runs through all the modules, and terminates at some specified point. For each module we also have a specified entering and leaving point for this test wiring. This problem also amounts to solving a Hamiltonian path problem with the difference that the distances are not symmetric and that starting and terminating point are specified. [2]

o The order-picking problem in warehouses.

This problem is associated with material handling in a warehouse (Ratliff & Rosenthal, 1983). Assume that at a warehouse an order arrives for a certain subset of the items stored in the warehouse. Some vehicle has to collect all items of this order to ship them to the customer. The relation to the TSP is immediately seen. The storage locations of the items correspond to the nodes of the graph. The distance between two nodes is given by the time needed to move the vehicle from one location to the other. The problem of finding a shortest route for the vehicle with minimum pickup time can now be solved as a TSP. In special cases this problem can be solved easily, see (van Dal, 1992) for an extensive discussion and for references. [2]

o Vehicle routing.

Suppose that in a city nmail boxes have to be emptied every day within a certain period of time, say 1 hour. The problem is to find the minimum number of trucks to do this and the shortest time to do the collections using this number of trucks. As another example, suppose that n customers require certain amounts of some commodities and a supplier has to satisfy all demands with a fleet of trucks. The problem isto find an assignment of customers to the trucks and a delivery schedule for each truck so that the capacity of each truck is not exceeded and the total travel distance is minimized. Several variations of these two problems, where time and capacity constraints are combined, are common in many real-world applications. This problem is solvable as a TSP if there are no time and capacity constraints and if the number of trucks is fixed (say m). In this case we obtain an m-salesmen problem. Nevertheless, one may apply methods for the TSP to find good feasible solutions for this problem (see Lenstra & Rinnooy Kan, 1974). [2]

o Mask plotting in PCB production.

For the production of each layer of a printed circuit board, as well as for layers of integrated semiconductor devices, a photographic mask has to be produced. In our case for printed circuit boards this is done by a mechanical plotting device. The plotter moves a lens over a photosensitive coated glass plate. The shutter may be opened or closed to expose specific parts of the plate. There are different apertures available to be able to generate different structures on the board. Two types of structures have to be considered. A line is exposed on the plate by moving the closed shutter to one end point of the line, then opening the shutter and moving it to the other endpoint of the line. Then the shutter isclosed. A point type structure is generated by moving (with the appropriate aperture) to the position of that point then opening the shutter just to make a short flash, and then closing it again. Exact modeling of the plotter control problem leads to a problem more complicated than the TSP and also more complicated than the rural postman problem. A real-world application in the actual production environment is reported in (Grötschel et al., 1991). [2]

4.1.3. Asymmetric and symmetric.

In the *symmetric TSP*, the distance between two cities is the same in each opposite direction, forming an undirected graph. This symmetry halves the number of possible solutions. In the *asymmetric TSP*, paths may not exist in both directions or the distances might be different, forming a directed graph. Traffic collisions, one-way streets, and airfares for cities with different departure and arrival fees are examples of how this symmetry could break down. [3]

4.1.4. Integer linear programming formulation.

TSP can be formulated as an integer linear program. Label the cities with the numbers o,..., n and define:

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

For *i* = o,..., *n*, let u_i be an artificial variable, and final taque c_{ij} to be the distance from city *i* to city *j*. Ten TSP can be griten as the following integer linear programming problema

$$\begin{aligned}
&\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} \\
&0 \leq x_{ij} \leq 1 && i, j = 0, \cdots, n \\
&u_i \in \mathbf{Z} && i = 0, \cdots, n \\
&\sum_{i=0, i \neq j}^{n} x_{ij} = 1 && j = 0, \cdots, n \\
&\sum_{j=0, j \neq i}^{n} x_{ij} = 1 && i = 0, \cdots, n \\
&u_i - u_j + n x_{ij} \leq n - 1 && 1 \leq i \neq j \leq n
\end{aligned}$$

The first set of equalities requires that each city be arrived at from exactly one other city, and the second set of equalities requires that from each city there is a departure to exactly one other city. The last constraints enforce that there is only a single tour covering all cities, and not two or more disjointed tours that only collectively cover all cities. To prove this, it is shown below (1) that every feasible solution contains only one closed sequence of cities, and (2) that for every single tour covering all cities, there are values for the dummy variables u_i that satisfy the constraints.

To prove that every feasible solution contains only one closed sequence of cities, it suffices to show that every subtour in a feasible solution passes through city 0 (noting $x_{ij} = 1$ that the equalities ensure there can only be one such tour). For if we sum all the inequalities corresponding to for any subtour of *k* steps not passing through city 0, we obtain:

$nk \leq (n-1)k,$ Which is a contradiction.

It now must be shown that for every single tour covering all cities, there are values for the dummy variables u_i that satisfy the constraints.

Without loss of generality, define the tour as originating (and ending) at city 0. Choose $u_i = t$ if city *i* is visited in step *t* (*i*, *t* = 1, 2, ... n). Then:

$$u_i - u_j \leq n - 1,$$

u_i

Since $x_{ij} = 0.$ $x_{ij} = 1$ can be no greater than *n* and can be no less than 1; hence the constraints are satisfied whenever For, we have:

$$u_i - u_j + nx_{ij} = (t) - (t+1) + n = n - 1,$$

Satisfying the constraint.

[4]

4.2. VEHICLE ROUTING PROBLEM – A TSP Generalization.

The VRP formulated as "The Truck Dispatching Problem" by G.B. Dantzig and J.H. Ramser in 1959 can be considered as a generalization of the "Traveling-Salesman Problem".
It is also combinatorial optimization and integer programming problem that was proposed for delivering gasoline from bulk terminal to service stations seeking to service a number of customers with a fleet of vehicles. [5]

4.2.1. Description.

VRP is an important problem in the fields of transportation, distribution, and logistics. Often, the context is that of delivering goods located at a central depot to customers who have ordered an specific quantity of such goods. The objective of the VRP is to minimize the total route cost satisfying each demand.

The total number of different routes through n cities is ½ n! Even for small values of n the total number of routes is exceedingly large, e.g. for n = 15, there are 653.837.184.000 different routes. [6]

4.2.1. Problem classification.

- Vehicle Routing Problem with Pickup and Delivery (VRPPD): A number of goods need to be moved from certain pickup locations to other delivery locations. The goal is to find optimal routes for a fleet of vehicles to visit the pickup and drop-off locations.

- Vehicle Routing Problem with LIFO: Similar to the VRPPD, except an additional restriction is placed on the loading of the vehicles: at any delivery location, the item being delivered must be the item most recently picked up. This scheme reduces the loading and unloading times at delivery locations because there is no need to temporarily unload items other than the ones that should be dropped off.

- Vehicle Routing Problem with Time Windows (VRPTW): The delivery locations have time windows within which the deliveries (or visits) must be made.

- Capacitated Vehicle Routing Problem (with or without Time Windows): CVRP or CVRPTW. The vehicles have limited carrying capacity of the goods that must be delivered.
- Vehicle Routing Problem with Multiple Trips (VRPMT): The vehicles can do more than one route.
- Open Vehicle Routing Problem (OVRP): Vehicles are not required to return to the depot.

[7]

4.2.2. Capacitated Multiple TSP Formulation with Lingo. – Example.

An important practical problem is the routing of vehicles from a central depot. An example is the routing of delivery trucks for a metropolitan newspaper. You can think of this as a multiple traveling salesperson problem with finite capacity for each salesperson. This problem is sometimes called the LTL (Less than TruckLoad) routing problem because a typical recipient receives less than a truck load of goods.

Given:

V = capacity of a vehicle
d_j = demand of city or stop j

Each city, j, must be visited once for $j > 1$:

$$\sum_i x_{ij} = 1$$

Each city $i > 1$, must be exited once:

$$\sum_i x_{ij} = 1$$

No subtours:

$$\sum_{i,j \in s} x_{ij} \leq |S| - 1,$$

No overloads: For each set of cities T, including 1, which constitute more than a truckload:

$$\sum_{i,j \in T} x_{ij} \leq |T| - k,$$

Where k = minimum number of cities that must be dropped from T to reduce it to one load.

```
MODEL! (VROUTE);
```

! The Vehicle Routing Problem (VRP) occurs in many service systems such as delivery, customer pick-up, repair and maintenance. A fleet of vehicles, each with fixed capacity, starts at a common depot and returns to the depot after visiting locations where service is demanded. Problems with more than a dozen cities can take lots of time.

This instance involves delivering the required amount of goods to 9 cities from a depot at city 1;

SETS: CITY/ Chi Den Frsn Hous KC LA Oakl Anah Peor Phnx/: Q, U;

! Q(I) = amount required at city I(given), must be delivered by just 1 vehicle.

U(I) = accumulated deliveries at city I ;

CXC(CITY, CITY): DIST, X;

! DIST(I,J) = distance from city I to city J

X(I,J) is 0-1 variable,

= 1 if some vehicle travels from city I to J,

else 0 ;

ENDSETS DATA:

! city 1 represents the common depot, i.e. Q(1) = 0;

Q= 0 6 3 7 7 18 4 5 2 6;

! distance from city I to city J is same from J to I, distance from city I to the depot is 0, because vehicle need not return to the depot ;

```
DIST=  ! To City;
!Chi  Den Frsn Hous   KC  LA Oakl Anah  Peor Phnx  From;
  0  996 2162 1067  499 2054 2134 2050  151 1713!  Chicago;
  0    0 1167 1019  596 1059 1227 1055  904  792!  Denver;
  0 1167    0 1747 1723  214  168  250 2070  598!  Fresno;
  0 1019 1747    0  710 1538 1904 1528  948 1149!  Houston;
  0  596 1723  710    0 1589 1827 1579  354 1214!  K. City;
  0 1059  214 1538 1589    0  371   36 1943  389!  L. A.;
  0 1227  168 1904 1827  371    0  407 2043  755!  Oakland;
  0 1055  250 1528 1579   36  407    0 1933  379!  Anaheim;
  0  904 2070  948  354 1943 2043 1933    0 1568!  Peoria;
  0  792  598 1149 1214  389  755  379 1568    0;! Phoenix;
```

! VCAP is the capacity of a vehicle;

VCAP = 18;

ENDDATA

```
! The objective is to minimize total travel distance;
MIN = @SUM( CXC: DIST * X);
! for each city, except depot....; @FOR( CITY( K)| K #GT# 1:
! a vehicle does not travel inside itself,...;
X( K, K) = 0;
! a vehicle must enter it,... ;
@SUM( CITY( I)| I #NE# K #AND# ( I #EQ# 1 #OR#
Q( I) + Q( K) #LE# VCAP): X( I, K)) = 1;
! a vehicle must leave it after service ;
@SUM( CITY( J)| J #NE# K #AND# ( J #EQ# 1 #OR#
Q( J) + Q( K) #LE# VCAP): X( K, J)) = 1;
! U( K) = amount delivered on trip up to city K
>= amount needed at K but <= vehicle capacity;
@BND( Q( K), U( K), VCAP);
! If K follows I, then can bound U( K) - U( I);
@FOR( CITY( I)| I #NE# K #AND# I #NE# 1: U( K) >= U( I) + Q( K) - VCAP + VCAP*( X(
K, I) + X( I, K))
- ( Q( K) + Q( I)) * X( K, I);
) ;
! If K is 1st stop, then U( K) = Q( K);
U( K) <= VCAP - ( VCAP - Q( K)) * X( 1, K);
! If K is not 1st stop...;
U( K) >=
Q( K)+ @SUM( CITY( I)| I #GT# 1: Q( I) * X( I, K));
) ;
! Make the X's binary;
@FOR( CXC( I, J): @BIN( X( I, J)) ;);
! Must send enough vehicles out of depot;
@SUM( CITY( J)| J #GT# 1: X( 1, J)) >=
@FLOOR((@SUM( CITY( I)| I #GT# 1: Q( I))/ VCAP) + .999);
```

END

```
Optimal solution found at step:       973
 Objective value:                6732.000

                     Variable           Value
                X( CHI, HOUS)        1.000000
                  X( CHI, LA)        1.000000
                X( CHI, PEOR)        1.000000
                X( CHI, PHNX)        1.000000
                 X( DEN, CHI)        1.000000
               X( FRSN, OAKL)        1.000000
                X( HOUS, CHI)        1.000000
                  X( KC, DEN)        1.000000
                  X( LA, CHI)        1.000000
                X( OAKL, CHI)        1.000000
               X( ANAH, FRSN)        1.000000
                 X( PEOR, KC)        1.000000
               X( PHNX, ANAH)        1.000000
```

By following the links, you can observe that the trips are:

Chicago - Houston;
Chicago - LA;
Chicago - Peoria - KC - Denver;
Chicago - Phoenix - Anaheim - Fresno - Oakland.

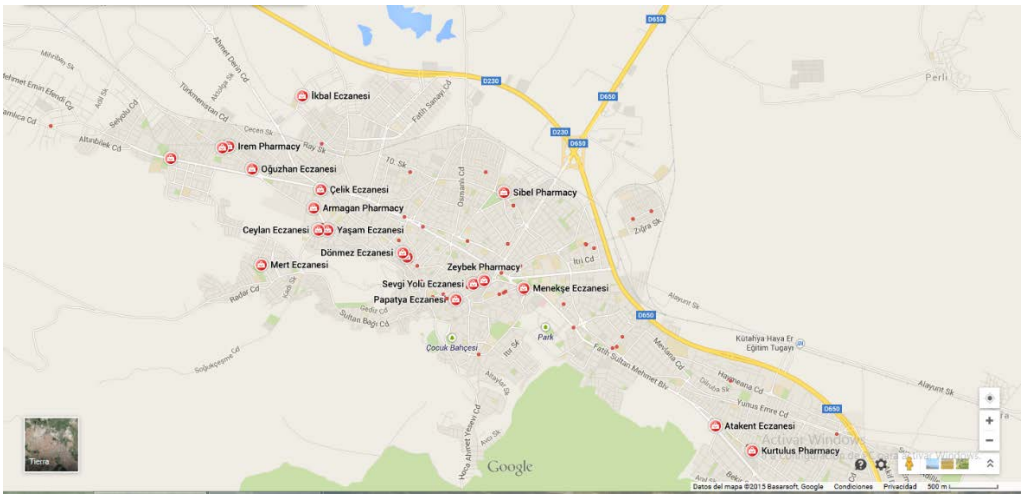
[8]

5. PROJECT DEVELOPMENT.

5.1. DATA AQUIRING.

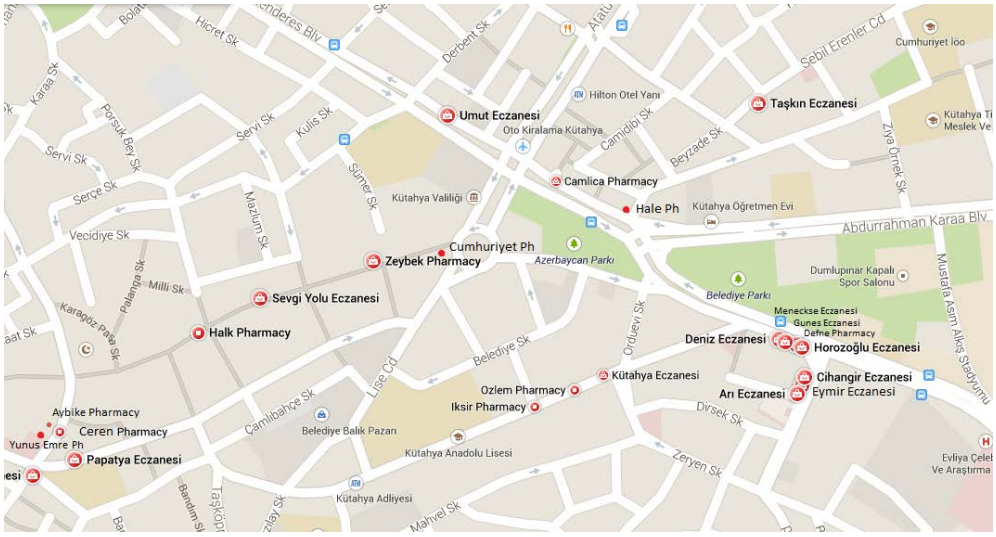
First of all is to place all the cities in a map (pharmacies in our project). For this task Google Maps or other online map can be used. Then is time to check that all the pharmacies are where they are supposed to be and enssure they are functioning nowadays.

Typing "Pharmacies & Eczanesi Kutahya" in Google Maps a map with all the pharmacies in Kutahya city is obtained:



[9]

Google find around 100 pharmacies in Kutahya city but this Project is comprising just the city center (the area close to the intersection between "Ataturk Blv" and "Menderes Blv").



[10]

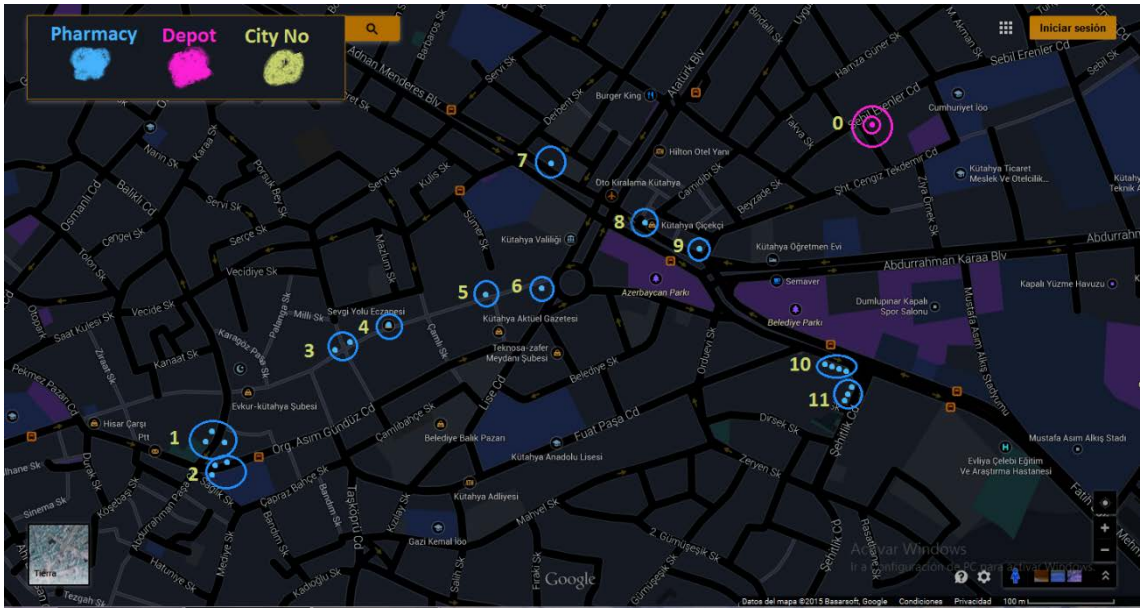
After checking all the pharmacies by walk, was found that some pharmacies had closed long time ago or just did not exist in the place which Google Maps indicated.

On the other hand, some of the pharmacies existing in the area were placed manually due to they were not uploaded to Google Maps.

A) ***Closed time ago or not existing***: "Iksir", "Ozlem", "Kutahya", "Deniz", "Taskin" and "Eymir".

B) ***Not uploaded to the system***: "Aybike", "Yunus Emre", "Cumhuriyet" and "Hale".

After applying this changes we proceed to group pharmacies which are really close ones from each others in order to not complicate the problem unneccesarily. Each group is a city. In the Screenshot we can see 12 cities cause Depot is the first city.



[11]

City No → Pharmacies

0	Depot
1	Aybike+Yunus+Ceren
2	Esra Eczane&Optic + Papatya + Carsi Zeytinler
3	Halk + Sifa
4	Sevgi Yolu
5	Zeybek
6	Cumhuriyet
7	Umut
8	Camlica
9	Hale
10	Gunes+Menekse+Defne+Horozoglu
11	Ari+Yigit+Cihangir

5.1.1. Distance and Time Matrixes.

Using the tool << How to go from "Point A" till "Point B" >> we obtain all the times and distances among the different cities, as it is shown below:

	0 Depot	1 Aybike+Yunus+Ceren	2 Eczane&Optic + Papatya + Carsi Zey	3 Halk + Sifa	4 SevgiYolu	5 Zeybek	6 Cumhuriyet	7 Umut	8 Camlica	9 Hale	10 Gunes+Menekse+DefneHorozoglu	11 Ari+Yigit+Cihangir
0	Depot	0	1400;5	1500;5	1400;4.2	1100;4.1	1000;4	1200;4.1	650;3	500;2	400;1.9	800;3	850;3.1
1	Aybike+Yunus+Ceren	1100;4	0	76;1	1200;5	1100;4	1000;4	600;3	750;3	900;4	850.4	550;2	950;4
2	Esra Eczane&Optic + Papatya + Carsi Zeytinler	1000;4	1300;5	0	1100;4	1000;4	900;3	550;2	650;2	850;3	750;3	800;3	850;3
3	Halk + Sifa	1000;5	450;2	500;2	0	68;1	350;1	600;3	650;3	800;4	750;4	850;4	950;4
4	SevgiYolu	999;4.9	450;1	550;2	68;1	0	550;3	650;3	700;3	850;4	800.4	900;4	1000;4
5	Zeybek	950;4	1000;4	700;3	350;1	300;1	0	600;3	600;3	800;4	700;3	850;4	900;4
6	Cumhuriyet	600;3	900;3	1000;4	650;3	550;2	500;2	0	280;1	400;2	350;2	350;1	450;2
7	Umut	600;3	650;2	750;3	400;2	300;1	230;1	280;1	0	450;2	400;2	500;3	600;3
8	Camlica	450;2	800;3	850;3	550;2	450;2	350;2	170;1	130;1	0	290;2	450;2	500;2
9	Hale	300;1	850;3	950;4	600;2	500;2	450;2	240;1	200;1	76;1	0	500;2	550;2
10	Gunes+Menekse+Defne+Horozoglu	600;2	1200;4	1200;5	900;4	800;3	700;3	550;2	500;2	350;1	290;1	0	64;1
11	Ari+Yigit+Cihangir	601;3	1201;5	1200;5	950;4	850;4	750;3	550;3	500;2	400;2	300;2	450;2	0

*Rows indicates"from" and colomns "indicate" untill.

According with the matrix it is observed that our problem is asymetrical since A→B time and distance generally is different than B→A. This is due to one-way streets.

5.1.2. Demand Assignment.

So as to learn which type of vehicle and whether more than one are needed, we asked pharmacy managers about the total amount of weight they order and the frequency they do it. As a result, they gave us an upward estimation which is shown in graphic 3.

Some pharmacies close to hospital or shopping streets sometimes have to make orders even all days owing to the high demand. Because of this, the delibery period must not be longer than weekly.

City demand in Kilograms a week.

1	Aybike+Yunus+Ceren	141
2	Esra Eczane&Optic + Papatya + Carsi Zeytinler	136
3	Halk + Sifa	81
4	Sevgi Yolu	45
5	Zeybek	50
6	Cumhuriyet	50
7	Umut	45
8	Camlica	55
9	Hale	62.5
10	Gunes+Menekse+Defne+Horozoglu	128
11	Ari+Yigit+Cihangir	130
	TOTAL	923.5

The total load is about 920 kg so one van will be enough for the deliver.

5.2. SELECTING AN ADECUATE ALGORITHM.

As the number os cities is not high, an exact algorithm is the best option since it allows to get the optimal solution. For solving our problem, we have used "Lingo 15.0 educational version software" and taken advantaje of its VRP model. As finally just one vehicle is needed, our problem gets simplified into a traditional Travelling-Salesman problem. Anyway, we are using VRP algorithm in case that demand gets increased and an extra vehicle may be needed.

We are going to execute the algorithm twice. First time for minimizing total distance and second time for minimizing total time.

5.3. DISTANCE MINIMIZATION.

5.3.1. Implementing algorithm for our data in Lingo (Distance minimization).

```
 MODEL:

! The Vehicle Routing Problem (VRP);

!***********************************;
! WARNING: Runtimes for this model   ;
! increase dramatically as the number;
! of cities increase. Formulations   ;
! with more than a dozen cities      ;
! WILL NOT SOLVE in a reasonable     ;
! amount of time!                    ;
!***********************************;

 SETS:
  ! Q(I) is the amount required at city I,
    U(I) is the accumulated delivers at city I ;
   CITY/1..12/: Q, U;

  ! DIST(I,J) is the distance from city I to city J
    X(I,J) is 0-1 variable: It is 1 if some vehicle
    travels from city I to J, 0 if none;
   CXC( CITY, CITY): DIST, X;
 ENDSETS

 DATA:
  ! city 0 represent the common depot;
   Q  =  0 141 136 81 45 50 50 45 55 62.5 128 130;
```

```
! distance from city I to city J is same from city

   J to city I distance from city I to the depot is
   0, since the vehicle has to return to the depot;



   DIST =  ! To City;
 !0   1     2     3     4     5     6     7     8     9     10    11 from;
  0   1400  1500  1400  1100  1000  1200  650   500   400   800   850!0;
1100  0     76    1200  1100  1000  600   750   900   850   550   950!1;
1000  1300  0     1100  1000  900   550   650   850   750   800   850!2;
1000  450   500   0     68    350   600   650   800   750   850   950!3;
999   450   550   68    0     550   650   700   850   800   900   1000!4;
950   1000  700   350   300   0     600   600   800   700   850   900!5;
600   900   1000  650   550   500   0     280   400   350   350   450!6;
600   650   750   400   300   230   260   0     450   400   500   600!7;
450   800   850   550   450   350   170   130   0     290   450   500!8;
300   850   950   600   500   450   240   200   76    0     500   550!9;
600   1200  1200  900   800   700   550   500   350   290   0     64!10;
601   1201  1200  950   850   750   550   500   400   300   450   0;!11;

! VCAP is the capacity of a vehicle ;
   VCAP = 1000;
 ENDDATA

  ! Minimize total travel distance;
   MIN = @SUM( CXC: DIST * X);

  ! For each city, except depot....;
   @FOR( CITY( K)| K #GT# 1:

  ! a vehicle does not traval inside itself,...;
     X( K, K) = 0;

  ! a vehicle must enter it,... ;
     @SUM( CITY( I)| I #NE# K #AND# ( I #EQ# 1 #OR#
      Q( I) + Q( K) #LE# VCAP): X( I, K)) = 1;

  ! a vehicle must leave it after service ;
     @SUM( CITY( J)| J #NE# K #AND# ( J #EQ# 1 #OR#
      Q( J) + Q( K) #LE# VCAP): X( K, J)) = 1;

  ! U( K) is at least amount needed at K but can't
    exceed capacity;
     @BND( Q( K), U( K), VCAP);

  ! If K follows I, then can bound U( K) - U( I);
     @FOR( CITY( I)| I #NE# K #AND# I #NE# 1:
      U( K) >= U( I) + Q( K) - VCAP + VCAP *
       ( X( K, I) + X( I, K)) - ( Q( K) + Q( I))
        * X( K, I);
     );
```

```
! If K is 1st stop, then U( K) = Q( K);
   U( K) <= VCAP - ( VCAP - Q( K)) * X( 1, K);

! If K is not 1st stop...;
   U( K)>= Q( K)+ @SUM( CITY( I)|
    I #GT# 1: Q( I) * X( I, K));
 );

! Make the X's binary;
 @FOR( CXC: @BIN( X));

! Minimum no. vehicles required, fractional
  and rounded;
 VEHCLF = @SUM( CITY( I)| I #GT# 1: Q( I))/ VCAP;



 VEHCLR = VEHCLF + 1.999 -
  @WRAP( VEHCLF - .001, 1);

! Must send enough vehicles out of depot;
 @SUM( CITY( J)| J #GT# 1: X( 1, J)) >= VEHCLR;
END
```

[12]

5.3.2. Obtaining the optimal solution with lingo (Distance minimization).

```
Global optimal solution found.
Objective value:                              3295.000
Objective bound:                              3295.000
Infeasibilities:                              0.000000
Extended solver steps:                               0
Total solver iterations:                            44
Elapsed runtime seconds:                          0.08

Model Class:                                      MILP

Total variables:                   145
Nonlinear variables:                 0
Integer variables:                 133

Total constraints:                 156
Nonlinear constraints:               0

Total nonzeros:                    968
Nonlinear nonzeros:                  0


  Variable           Value        Reduced Cost

  VCAP            1000.000            0.000000
  VEHCLF          0.9235000           0.000000
  VEHCLR          1.000000            0.000000
  Q( 1)           0.000000            0.000000
  Q( 2)           141.0000            0.000000
  Q( 3)           136.0000            0.000000
  Q( 4)           81.00000            0.000000
  Q( 5)           45.00000            0.000000
  Q( 6)           50.00000            0.000000
```

```
Q( 7)        50.00000          0.000000
Q( 8)        45.00000          0.000000
Q( 9)        55.00000          0.000000
Q( 10)       62.50000          0.000000
Q( 11)       128.0000          0.000000
Q( 12)       130.0000          0.000000
U( 1)        0.000000          0.000000
U( 2)        479.5000          0.000000
U( 3)        615.5000          0.000000
U( 4)        338.5000          0.000000
U( 5)        257.5000          0.000000
U( 6)        212.5000          0.000000
U( 7)        665.5000          0.000000
U( 8)        162.5000          0.000000
U( 9)        117.5000          0.000000
U( 10)       62.50000          0.000000
U( 11)       793.5000          0.000000
U( 12)       923.5000          0.000000


X( 1, 1)       1.000000          0.000000
X( 1, 2)       0.000000          1400.000
X( 1, 3)       0.000000          1500.000
X( 1, 4)       0.000000          1400.000
X( 1, 5)       0.000000          1100.000
X( 1, 6)       0.000000          1000.000
X( 1, 7)       0.000000          1200.000
X( 1, 8)       0.000000          650.0000
X( 1, 9)       0.000000          500.0000
X( 1, 10)      1.000000          400.0000
X( 1, 11)       0.000000          800.0000
X( 1, 12)       0.000000          850.0000
X( 2, 1)       0.000000          1100.000
X( 2, 2)       0.000000          0.000000
X( 2, 3)       1.000000          76.00000
X( 2, 4)       0.000000          1200.000
X( 2, 5)       0.000000          1100.000
X( 2, 6)       0.000000          1000.000
X( 2, 7)       0.000000          600.0000
X( 2, 8)       0.000000          750.0000
X( 2, 9)       0.000000          900.0000
X( 2, 10)       0.000000          850.0000
X( 2, 11)       0.000000          550.0000
X( 2, 12)       0.000000          950.0000
X( 3, 1)       0.000000          1000.000
X( 3, 2)       0.000000          1300.000
X( 3, 3)       0.000000          0.000000
X( 3, 4)       0.000000          1100.000
X( 3, 5)       0.000000          1000.000
X( 3, 6)       0.000000          900.0000
X( 3, 7)       1.000000          550.0000
X( 3, 8)       0.000000          650.0000
X( 3, 9)       0.000000          850.0000
X( 3, 10)       0.000000          750.0000
X( 3, 11)       0.000000          800.0000
X( 3, 12)       0.000000          850.0000
X( 4, 1)       0.000000          1000.000
X( 4, 2)       1.000000          450.0000
X( 4, 3)       0.000000          500.0000
```

X(4, 4)	0.000000	0.000000
X(4, 5)	0.000000	68.00000
X(4, 6)	0.000000	350.0000
X(4, 7)	0.000000	600.0000
X(4, 8)	0.000000	650.0000
X(4, 9)	0.000000	800.0000
X(4, 10)	0.000000	750.0000
X(4, 11)	0.000000	850.0000
X(4, 12)	0.000000	950.0000
X(5, 1)	0.000000	999.0000
X(5, 2)	0.000000	450.0000
X(5, 3)	0.000000	550.0000
X(5, 4)	1.000000	68.00000
X(5, 5)	0.000000	0.000000
X(5, 6)	0.000000	550.0000
X(5, 7)	0.000000	650.0000
X(5, 8)	0.000000	700.0000
X(5, 9)	0.000000	850.0000
X(5, 10)	0.000000	800.0000
X(5, 11)	0.000000	900.0000
X(5, 12)	0.000000	1000.000
X(6, 1)	0.000000	950.0000
X(6, 2)	0.000000	1000.000
X(6, 3)	0.000000	700.0000
X(6, 4)	0.000000	350.0000
X(6, 5)	1.000000	300.0000
X(6, 6)	0.000000	0.000000
X(6, 7)	0.000000	600.0000
X(6, 8)	0.000000	600.0000
X(6, 9)	0.000000	800.0000
X(6, 10)	0.000000	700.0000
X(6, 11)	0.000000	850.0000
X(6, 12)	0.000000	900.0000
X(7, 1)	0.000000	600.0000
X(7, 2)	0.000000	900.0000
X(7, 3)	0.000000	1000.000
X(7, 4)	0.000000	650.0000
X(7, 5)	0.000000	550.0000
X(7, 6)	0.000000	500.0000
X(7, 7)	0.000000	0.000000
X(7, 8)	0.000000	280.0000
X(7, 9)	0.000000	400.0000
X(7, 10)	0.000000	350.0000
X(7, 11)	1.000000	350.0000
X(7, 12)	0.000000	450.0000
X(8, 1)	0.000000	600.0000
X(8, 2)	0.000000	650.0000
X(8, 3)	0.000000	750.0000
X(8, 4)	0.000000	400.0000
X(8, 5)	0.000000	300.0000
X(8, 6)	1.000000	230.0000
X(8, 7)	0.000000	260.0000
X(8, 8)	0.000000	0.000000
X(8, 9)	0.000000	450.0000
X(8, 10)	0.000000	400.0000
X(8, 11)	0.000000	500.0000
X(8, 12)	0.000000	600.0000

```
X( 9, 1)        0.000000            450.0000
X( 9, 2)        0.000000            800.0000
X( 9, 3)        0.000000            850.0000
X( 9, 4)        0.000000            550.0000
X( 9, 5)        0.000000            450.0000
X( 9, 6)        0.000000            350.0000
X( 9, 7)        0.000000            170.0000
X( 9, 8)        1.000000            130.0000
X( 9, 9)        0.000000            0.000000
X( 9, 10)       0.000000            290.0000
X( 9, 11)       0.000000            450.0000
X( 9, 12)       0.000000            500.0000
X( 10, 1)       0.000000            300.0000
X( 10, 2)       0.000000            850.0000
X( 10, 3)       0.000000            950.0000
X( 10, 4)       0.000000            600.0000
X( 10, 5)       0.000000            500.0000
X( 10, 6)       0.000000            450.0000
X( 10, 7)       0.000000            240.0000
X( 10, 8)       0.000000            200.0000
X( 10, 9)       1.000000            76.00000
X( 10, 10)      0.000000            0.000000
X( 10, 11)      0.000000            500.0000
X( 10, 12)      0.000000            550.0000
X( 11, 1)       0.000000            600.0000
X( 11, 2)       0.000000            1200.000
X( 11, 3)       0.000000            1200.000
X( 11, 4)       0.000000            900.0000
X( 11, 5)       0.000000            800.0000
X( 11, 6)       0.000000            700.0000
X( 11, 7)       0.000000            550.0000
X( 11, 8)       0.000000            500.0000
X( 11, 9)       0.000000            350.0000
X( 11, 10)      0.000000            290.0000
X( 11, 11)      0.000000            0.000000
X( 11, 12)      1.000000            64.00000
X( 12, 1)       1.000000            601.0000
X( 12, 2)       0.000000            1201.000
X( 12, 3)       0.000000            1200.000
X( 12, 4)       0.000000            950.0000
X( 12, 5)       0.000000            850.0000
X( 12, 6)       0.000000            750.0000
X( 12, 7)       0.000000            550.0000
X( 12, 8)       0.000000            500.0000
X( 12, 9)       0.000000            400.0000
X( 12, 10)      0.000000            300.0000
X( 12, 11)      0.000000            450.0000
X( 12, 12)      0.000000            0.000000
```

[13]

Following the sequence of xij=1 starting from depot, it is found that the optimal tour:

1 – 10 – 9 – 8 – 6 – 5 – 4 – 2 – 3 – 7 – 11 – 12 – 1

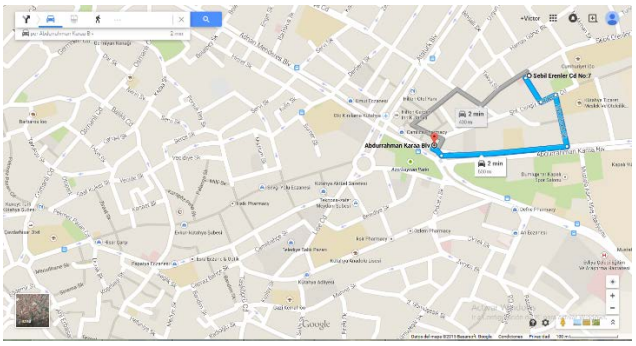
In the programming, depot is number 1 but in our Project we set it has number 0 so we just have to substract 1 to all the tour. Our optimal tour would be:

0 – 9 – 8 – 7 – 5 – 4 – 3 – 1 – 2 – 6 – 10 – 11 – 0

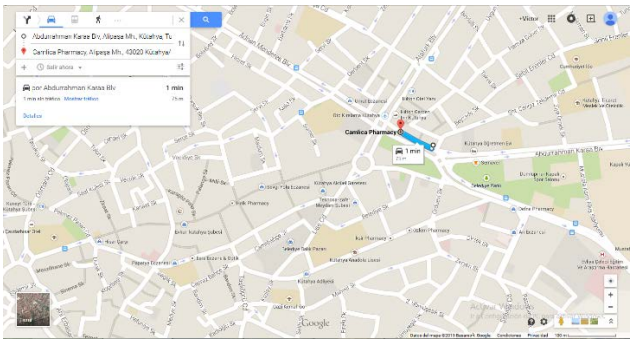
5.3.3. Graphical Checking (Distance minimization).

Let´s check graphically whether the tour seems optimal or it does not:

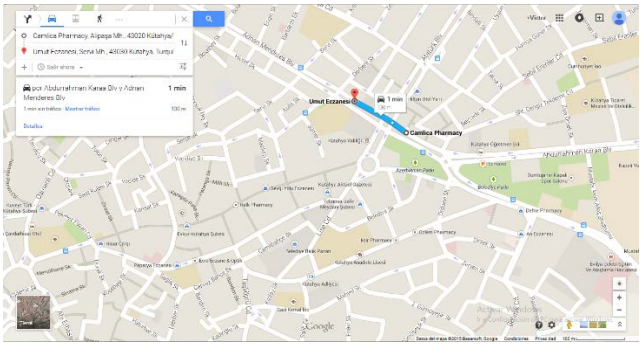
0 - 9



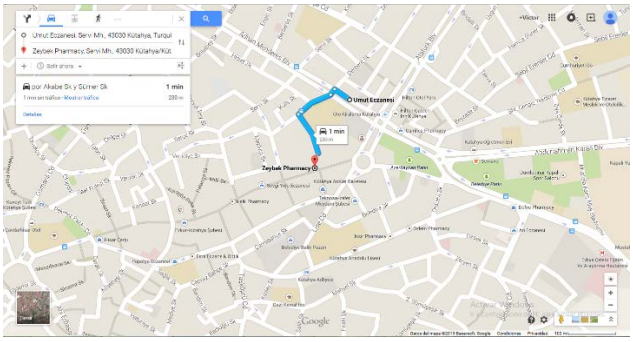
9 - 8



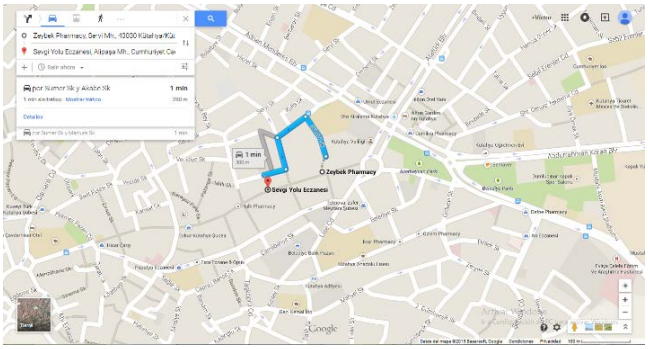
8 - 7



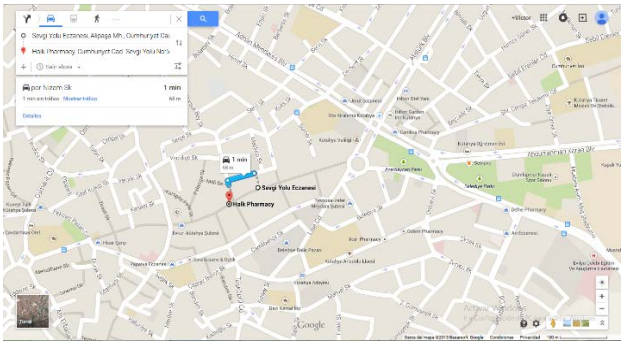
7 - 5



5 - 4

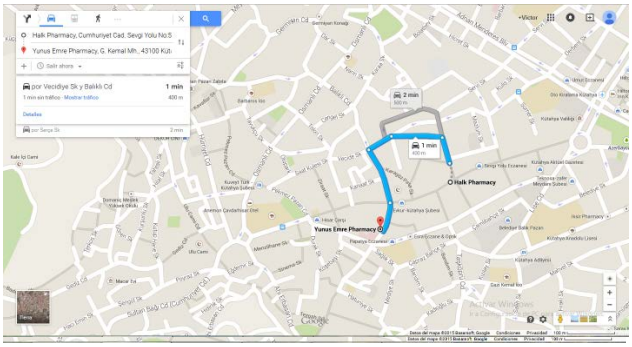


4 - 3



2 - 6

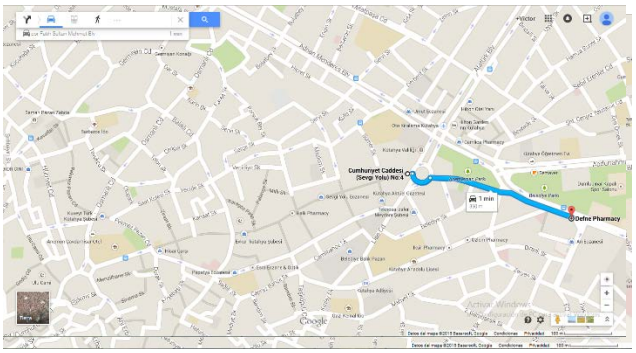
3 - 1



1 - 2

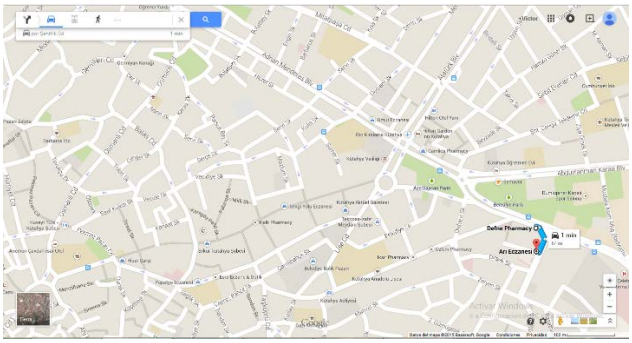
6 -10

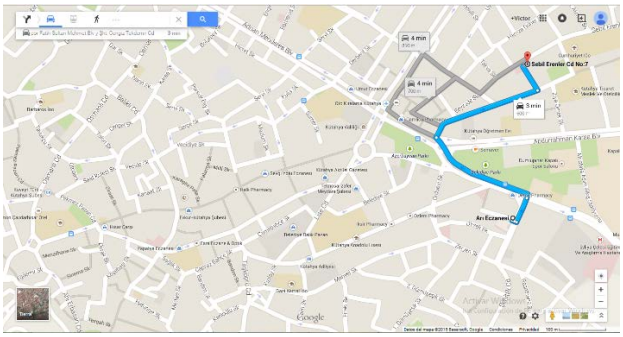
10 - 11



11 - 0

[14]





5.4. TIME MINIMIZATION.

5.4.1. Implementing algorithm for our data (Time minimization).

Same model is used. We just have to replace the distance matrix with the time matrix.

```
TIME =  ! To City;
!0   1    2    3    4    5    6    7    8    9    10   11 from;
 0   5    5    4.2  4.1  4    4.1  3    2    1.9  3    3.1 !0;
 4   0    1    5    4    4    3    3    4    4    2    4!1;
 4   5    0    4    4    3    2    2    3    3    3    3!2;
 5   2    2    0    1    1    3    3    4    4    4    4!3;
 4.9 1    2    1    0    3    3    3    4    4    4    4!4;
 4   4    3    1    1    0    3    3    4    3    4    4!5;
 3   3    4    3    2    2    0    1    2    2    1    2!6;
 3   2    3    2    1    1    1    0    2    2    3    3!7;
 2   3    3    2    2    2    1    1    0    2    2    2!8;
 1   3    4    2    2    2    1    1    1    0    2    2!9;
 2   4    5    4    3    3    2    2    1    1    0    1!10;
 3   5    5    4    4    3    3    2    2    2    2    0!11;
;
```

[15]

5.4.2. Obtaining the solution with lingo (Time minimization).

```
Global optimal solution found.
Objective value:                              15.00000
Objective bound:                              15.00000
Infeasibilities:                              0.000000
Extended solver steps:                               1
Total solver iterations:                           733
Elapsed runtime seconds:                          0.16

Model Class:                                      MILP

Total variables:                   145
Nonlinear variables:                 0
Integer variables:                 133

Total constraints:                 156
Nonlinear constraints:               0

Total nonzeros:                    968
Nonlinear nonzeros:                  0
```

Variable	Value	Reduced Cost
VCAP	1000.000	0.000000
VEHCLF	0.9235000	0.000000
VEHCLR	1.000000	0.000000
Q(1)	0.000000	0.000000
Q(2)	141.0000	0.000000
Q(3)	136.0000	0.000000
Q(4)	81.00000	0.000000
Q(5)	45.00000	0.000000
Q(6)	50.00000	0.000000
Q(7)	50.00000	0.000000
Q(8)	45.00000	0.000000
Q(9)	55.00000	0.000000
Q(10)	62.50000	0.000000
Q(11)	128.0000	0.000000
Q(12)	130.0000	0.000000
U(1)	0.000000	0.000000
U(2)	417.0000	0.000000
U(3)	553.0000	0.000000
U(4)	231.0000	0.000000
U(5)	276.0000	0.000000
U(6)	150.0000	0.000000
U(7)	603.0000	0.000000
U(8)	100.0000	0.000000
U(9)	55.00000	0.000000
U(10)	923.5000	0.000000
U(11)	731.0000	0.000000
U(12)	861.0000	0.000000
X(1, 1)	1.000000	0.000000
X(1, 2)	0.000000	5.000000
X(1, 3)	0.000000	5.000000
X(1, 4)	0.000000	4.200000
X(1, 5)	0.000000	4.100000
X(1, 6)	0.000000	4.000000
X(1, 7)	0.000000	4.100000
X(1, 8)	0.000000	3.000000
X(1, 9)	1.000000	2.000000
X(1, 10)	0.000000	1.900000
X(1, 11)	0.000000	3.000000
X(1, 12)	0.000000	3.100000
X(2, 1)	0.000000	4.000000
X(2, 2)	0.000000	0.000000
X(2, 3)	1.000000	1.000000
X(2, 4)	0.000000	5.000000
X(2, 5)	0.000000	4.000000
X(2, 6)	0.000000	4.000000
X(2, 7)	0.000000	3.000000
X(2, 8)	0.000000	3.000000
X(2, 9)	0.000000	4.000000
X(2, 10)	0.000000	4.000000
X(2, 11)	0.000000	2.000000
X(2, 12)	0.000000	4.000000
X(3, 1)	0.000000	4.000000
X(3, 2)	0.000000	5.000000
X(3, 3)	0.000000	0.000000
X(3, 4)	0.000000	4.000000
X(3, 5)	0.000000	4.000000
X(3, 6)	0.000000	3.000000
X(3, 7)	1.000000	2.000000

```
X( 3, 8)      0.000000           2.000000
X( 3, 9)      0.000000           3.000000
X( 3, 10)     0.000000           3.000000
X( 3, 11)     0.000000           3.000000
X( 3, 12)     0.000000           3.000000
X( 4, 1)      0.000000           5.000000
X( 4, 2)      0.000000           2.000000
X( 4, 3)      0.000000           2.000000
X( 4, 4)      0.000000           0.000000
X( 4, 5)      1.000000           1.000000
X( 4, 6)      0.000000           1.000000
X( 4, 7)      0.000000           3.000000
X( 4, 8)      0.000000           3.000000
X( 4, 9)      0.000000           4.000000
X( 4, 10)     0.000000           4.000000
X( 4, 11)     0.000000           4.000000
X( 4, 12)     0.000000           4.000000
X( 5, 1)      0.000000           4.900000
X( 5, 2)      1.000000           1.000000
X( 5, 3)      0.000000           2.000000
X( 5, 4)      0.000000           1.000000
X( 5, 5)      0.000000           0.000000
X( 5, 6)      0.000000           3.000000
X( 5, 7)      0.000000           3.000000
X( 5, 8)      0.000000           3.000000
X( 5, 9)      0.000000           4.000000
X( 5, 10)     0.000000           4.000000
X( 5, 11)     0.000000           4.000000
X( 5, 12)     0.000000           4.000000
X( 6, 1)      0.000000           4.000000
X( 6, 2)      0.000000           4.000000
X( 6, 3)      0.000000           3.000000
X( 6, 4)      1.000000           1.000000
X( 6, 5)      0.000000           1.000000
X( 6, 6)      0.000000           0.000000
X( 6, 7)      0.000000           3.000000
X( 6, 8)      0.000000           3.000000
X( 6, 9)      0.000000           4.000000
X( 6, 10)     0.000000           3.000000
X( 6, 11)     0.000000           4.000000
X( 6, 12)     0.000000           4.000000
X( 7, 1)      0.000000           3.000000
X( 7, 2)      0.000000           3.000000
X( 7, 3)      0.000000           4.000000
X( 7, 4)      0.000000           3.000000
X( 7, 5)      0.000000           2.000000
X( 7, 6)      0.000000           2.000000
X( 7, 7)      0.000000           0.000000
X( 7, 8)      0.000000           1.000000
X( 7, 9)      0.000000           2.000000
X( 7, 10)     0.000000           2.000000
X( 7, 11)     1.000000           1.000000
X( 7, 12)     0.000000           2.000000
X( 8, 1)      0.000000           3.000000
X( 8, 2)      0.000000           2.000000
X( 8, 3)      0.000000           3.000000
X( 8, 4)      0.000000           2.000000
X( 8, 5)      0.000000           1.000000
X( 8, 6)      1.000000           1.000000
X( 8, 7)      0.000000           1.000000
X( 8, 8)      0.000000           0.000000
```

```
X( 8, 9)      0.000000           2.000000
X( 8, 10)     0.000000           2.000000
X( 8, 11)     0.000000           3.000000
X( 8, 12)     0.000000           3.000000
X( 9, 1)      0.000000           2.000000
X( 9, 2)      0.000000           3.000000
X( 9, 3)      0.000000           3.000000
X( 9, 4)      0.000000           2.000000
X( 9, 5)      0.000000           2.000000
X( 9, 6)      0.000000           2.000000
X( 9, 7)      0.000000           1.000000
X( 9, 8)      1.000000           1.000000
X( 9, 9)      0.000000           0.000000
X( 9, 10)     0.000000           2.000000
X( 9, 11)     0.000000           2.000000
X( 9, 12)     0.000000           2.000000
X( 10, 1)     1.000000           1.000000
X( 10, 2)     0.000000           3.000000
X( 10, 3)     0.000000           4.000000
X( 10, 4)     0.000000           2.000000
X( 10, 5)     0.000000           2.000000
X( 10, 6)     0.000000           2.000000
X( 10, 7)     0.000000           1.000000
X( 10, 8)     0.000000           1.000000
X( 10, 9)     0.000000           1.000000
X( 10, 10)    0.000000           0.000000
X( 10, 11)    0.000000           2.000000
X( 10, 12)    0.000000           2.000000
X( 11, 1)     0.000000           2.000000
X( 11, 2)     0.000000           4.000000
X( 11, 3)     0.000000           5.000000
X( 11, 4)     0.000000           4.000000
X( 11, 5)     0.000000           3.000000
X( 11, 6)     0.000000           3.000000
X( 11, 7)     0.000000           2.000000
X( 11, 8)     0.000000           2.000000
X( 11, 9)     0.000000           1.000000
X( 11, 10)    0.000000           1.000000
X( 11, 11)    0.000000           0.000000
X( 11, 12)    1.000000           1.000000
X( 12, 1)     0.000000           3.000000
X( 12, 2)     0.000000           5.000000
X( 12, 3)     0.000000           5.000000
X( 12, 4)     0.000000           4.000000
X( 12, 5)     0.000000           4.000000
X( 12, 6)     0.000000           3.000000
X( 12, 7)     0.000000           3.000000
X( 12, 8)     0.000000           2.000000
X( 12, 9)     0.000000           2.000000
X( 12, 10)    1.000000           2.000000
X( 12, 11)    0.000000           2.000000
X( 12, 12)    0.000000           0.000000
```

[16]

Following the sequence of xij=1 starting from depot, it is found that the optimal tour is:

1 – 9 – 8 – 6 – 4 – 5 – 2 – 3 – 7 – 11 – 12 – 10 - 1

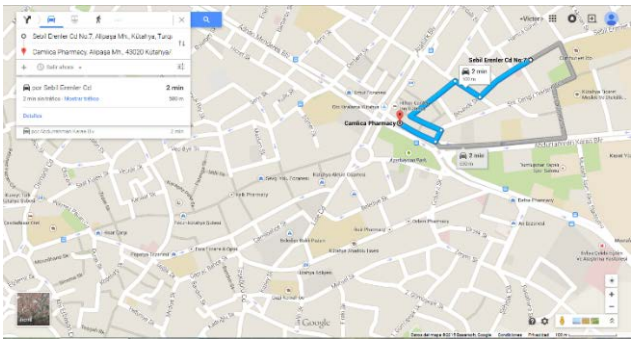
In the programming, number "1" indicates depot but in our project we set it as number "0" so we just have to substract "1" to all the tour. Our optimal tour would be:

0 – 8 – 7 – 5 – 3 – 4 – 1 – 2 – 6 – 10 – 11 – 9 – 0

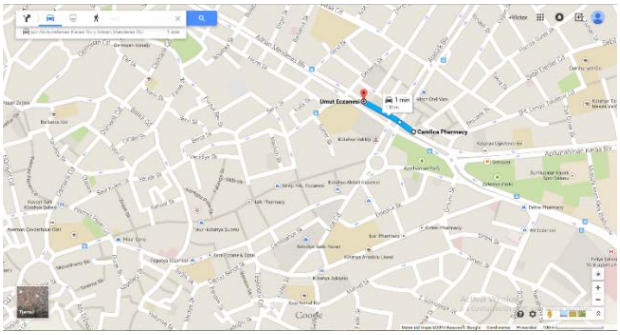
5.3.3. Graphical Checking (Time minimization).

Let´s check graphically whether the tour look like optimal or not:

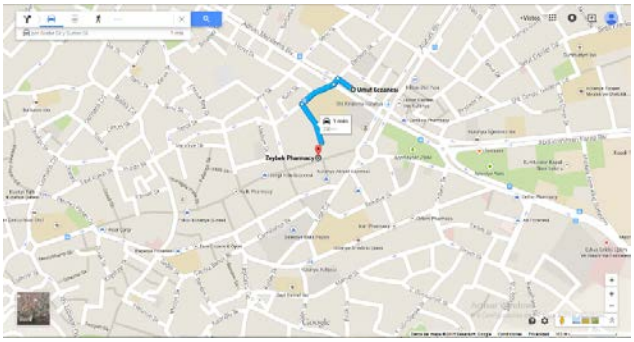
0 – 8



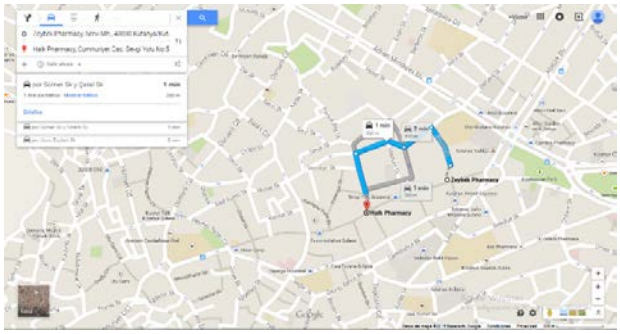
8 – 7



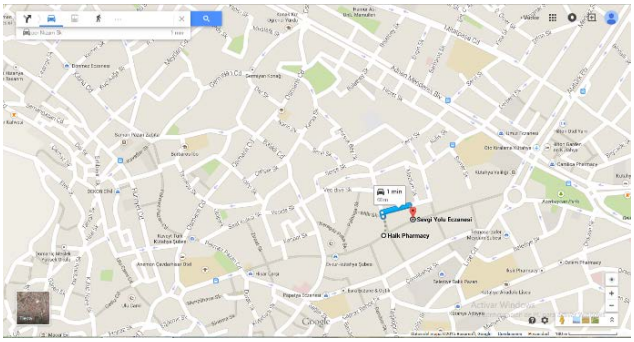
7 – 5

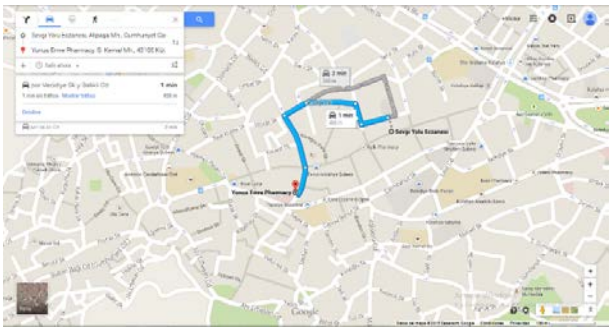


5 – 3 (*)



3 – 4 (*)

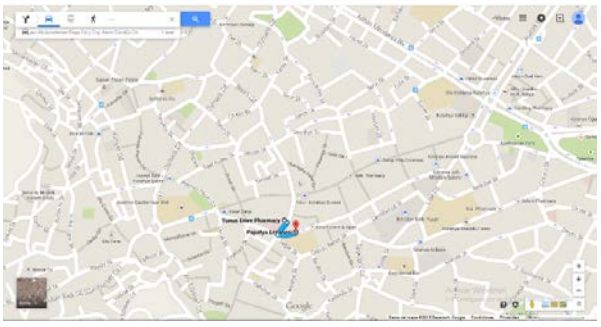


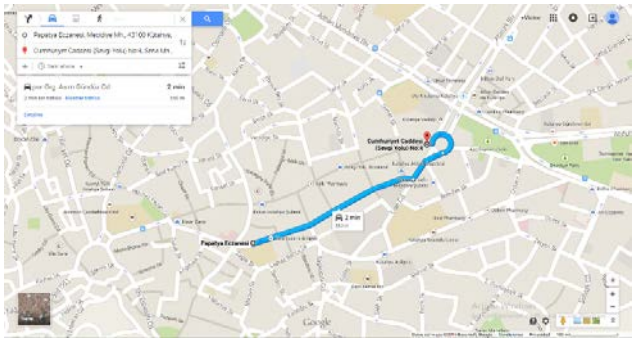


4 – 1 (*)

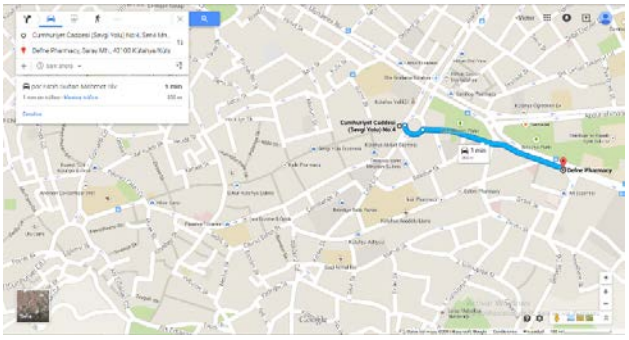
11 - 9

1 – 2





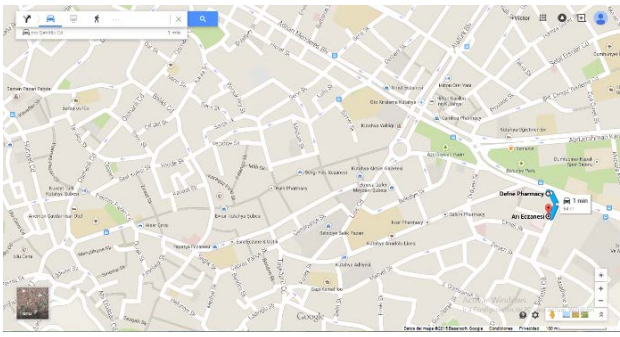
6 – 10



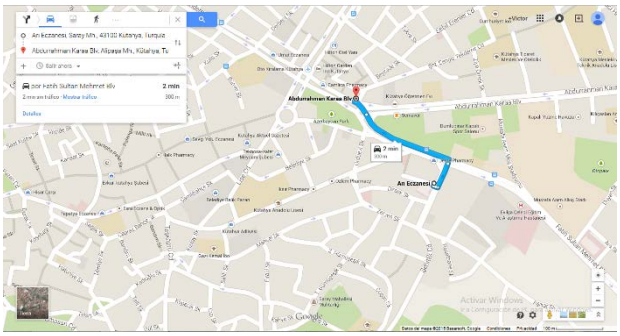
2 – 6

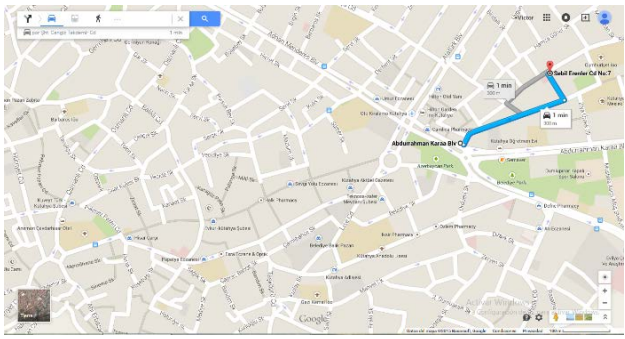
10 – 11

9 -



[17]





(***) Stretch"5 – 3 – 4 – 1" is incorrect. This is owed to the limited accuracy of the time matrix (maximum accuracy is 1 minute). Stretch 5 – 3 goes from "zeybeck" till "halk" instead of stopping in "sevgi yolu", which is closer, due to both paths are rounded to 1 minute.
This situation, causes an opposite direction in 3 – 4, making 4 – 1 longer because of passing from sevgi yolu to halk again.
Modifying this, the right stretch would be "5 – 4 – 3 – 1" which is the same as the obtained before in distance minimization.

So, after corrections, our optimal tour is:

0 – 8 – 7 – 5 – 4 – 3 – 1 – 2 – 6 – 10 – 11 – 9 – 0

5.5. TIME AND DISTANCE SOLUTION COMPARISON

Acording to distance min: **0 – 9 – 8 – 7 – 5 – 4 – 3 – 1 – 2 – 6 – 10 – 11 – 0**

Total distance = 400 + 75 + 130 + 230 + 300 + 68 + 400 + 50 + 550 + 350 + 64 + 600 = 3217 m

Total time = 2 + 1 + 1 + 1 + 1 + 0.5 + 1 + 0.5 + 2 + 1 + 0.5 + 3 = 14.5 minutes

---

Acording to time min: **0 – 8 – 7 – 5 – 4 – 3 – 1 – 2 – 6 – 10 – 11 – 9 – 0**

Total distance = 500 + 130 + 230 + 300 + 68 + 400 + 50 + 550 + 350 + 64 + 300 + 300 = 3241 m

Total time = 2 + 1 + 1 + 1 + 0.5 + 1 + 0.5 + 2 + 1 + 0.5 + 2 + 1 = 13.5 minutes

As we observe, the only difference between both solutions is that city 9 is visited on the way for distance minimization while it is visited on the return in time minimization case.

In distance minimization, total distance is 24 m less than in time minimization while in time minimization, time is 1 min less than in distance minimization, as it is supposed to be. So, it is conclude that both solutions are valid.

As it is apreciated, any of both routes can be considered optimal owing to their proximity in both total time and distance.

6. ECONOMICAL ANALYSIS.

Current cost

Need to know the transportation cost payed by each pharmacy a week. We could not obtain this data due to the unknowledge of the pharmacy or the lack of cooperation in some cases.

New Delivery Stage Cost.

One day van hire = 74 TL

P - CETVELİ
MOTORLU TAŞITLARIN ORTALAMA ALIM DEĞERLERİNİ VE SEFERBERLİK TATBİKATINA KATILACAK ARAÇLARIN GÜNLÜK KİRA BEDELLERİNİ GÖSTERİR CETVEL

(TL)

ARACIN CİNSİ VE TONAJI	ALIM DEĞERİ	SEFERBERLİK TATBİKATLARINA KATILACAK ARAÇLARIN TAHMİNİ GÜNLÜK KİRA BEDELİ
10 TON İDARİ ARAÇ	133.700	
5 TON İDARİ ARAÇ	72.000	----
1/2 TON İDARİ ARAÇ	24.000	----
1/5 TON TAKTİK ARAÇ	13.500	----
1/2 TON TAKTİK ARAÇ	21.400	----
5 TON TAKTİK ARAÇ	290.000	----
2.5 TON TAKTİK ARAÇ	227.000	----
10 TON ÇEKİCİ ARAÇ	879.400	----
KARÜSTÜ ARACI	392.300	----
KAMYONET 500 - 1000 KG.	----	74,00
KAMYONET 1001 - 2000 KG.	----	85,00
KAMYONET 2001 - 3000 KG.	----	102,00
KAMYON 3001 - 4000 KG.	----	112,00
KAMYON 4001 - 5000 KG.	----	137,00
KAMYON 5001 - 6000 KG.	----	137,00
KAMYON 6001 - 7000 KG.	----	137,00
KAMYON 7001 - 8000 KG.	----	164,00
KAMYON 8001 - 9000 KG.	----	164,00
KAMYON 9001- 10.000 KG.	----	164,00
KAMYON 10.000 - 11.000 KG.	----	164,00
KAMYON 11.001 - 12.000 KG.	----	178,00

[18]

Cost of renting a van for one day/21 pharmacies = 74 TL / 21 = **3.5 TL**

Fuel Price= 4.7 TL/L

Van consumption = 10 L / 100 Km

Total distance = 3.2 km / tour + 5 km extra = **8.2 km**

Gasoline cost = 3.85 / 21 = **0.18 TL**

One worker for the deliver + driving = 50 Tl / 21 = **2.38 TL**

Delivery stage cost = Renting + Gasoline + Worker = 3.5 + 0.2 + 2.38 = **6 TL a week each pharmacy**.

Total New Cost = Delivery stage cost + Collecting stage cost

Savings = Total Current Costs - Total New Costs

7. CONCLUSION.

Alhought pharmacies are a very dynamical system, statistical analysis of the demand would make possible weekly delivers which suppose important savings in transportation. On the other hand, sharing transportation costs, allows to reduce CO_2 emissions and other contaminants (NOx, VOCs, Ozone, CO, C_6H_6, PM_{10} and $PM_{2.5}$). A reduction about the 95% of these emmisions would be achieved if no extra orders were needed (best scenario).

However, still extra orders must be done when a peak demand comes in to a pharmacy. A good planification and statistic analysis by each pharmacy is needed to minimize this extra orders, but that is another case of study.

The Vehicle routing problem does not save an important amount of money in the delivery stage due to the high proximity of all the pharmacies, as it was known from the beginning. Nevertheless, the experience acquired will be pretty useful for the wholesaler collection stage in which the savings will be really important. Moreover, the algorithm can be used for solving this problem as long as the number of cities is not higher than approx 25, (in that case and Heuristic algorithm will be needed to find a non-optimal but good solution). The only things we have to change are the distance and time matrixes and the demands of each city, which might be higher because of keeping stock at depot. In this case more than one vehicle will be needed so our TSP will become a VPR with the same number of routes as vehicles.

For further studies, traffic situations such as rush hours and traffic lights should be taken into account. Also delivery times from van to city and city to van.

WEB LINKS AND REFERENCES.

[1]

http://en.wikipedia.org/wiki/Travelling_salesman_problem

[2]

http://cdn.intechopen.com/pdfs-wm/12736.pdf - 2. Applications and linkages

[3]

http://en.wikipedia.org/wiki/Travelling_salesman_problem#Asymmetric_and_symmetric

[4]

http://en.wikipedia.org/wiki/Travelling_salesman_problem#Integer_linear_programming_formulation

[5]

http://orbit.dtu.dk/fedora/objects/orbit:83400/datastreams/file_5280227/content

[6]

http://en.wikipedia.org/wiki/Vehicle_routing_problem

[7]

http://en.wikipedia.org/wiki/Vehicle_routing_problem - Overview

[8]

http://home.anadolu.edu.tr/~gurkan.o/VRP_lingo.pdf

[9]

https://www.google.com/maps/search/pharmacy+%26+Eczanesi+Kutahya/@39.416185,29.9870261,13z/data=!3m1!4b1

[10]

https://www.google.com/maps/search/pharmacy+%26+Eczanesi+Kutahya/@39.416185,29.9870261,13z/data=!3m1!4b1 - Zoom in.

[11]

Modification from a Google Maps screenshot.

[12]

Lingo 15.0 – Lingo Model – VROUTE Distances.

[13]

Lingo 15.0 – Solution Report – VROUTE Distances.

[14]

Google Maps sreenshots - Distances.

[15]

Lingo 15.0 – Lingo Model – VROUTE Times.

[16]

Lingo 15.0 – Solution Report – VROUTE Times..

[17]

Google Maps sreenshots - Times.

[18]

www.stratejigelistirme.itu.edu.tr

2015 YEAR OF CENTRAL GOVERNMENT BUDGET LAW

2015_YILI_MERKEZI_YONETIM_BUTCE_KANUNU.pdf - page 53