041500000 Doğa Poyraz Tahan
041503034 Modeling and Methods in Optimization Eren Melih Altun
041503059 Gülce Eker

P-Median Problem

P-Median problem is a specific type of a discrete location model. In this model, we want to place p facilities to minimize the demand-weighted distance between a demand node and the location in which a facility was placed.

Min

$$\sum_i \sum_j h_i d_{ij} Y_{ij} \qquad \text{(a)}$$

Subject to

$$\sum_j Y_{ij} = 1 \qquad \forall i \qquad \text{(b)}$$

$$\sum_j X_j = P \qquad \text{(c)}$$

$$Y_{ij} - X_j \leq 0 \qquad \forall i,j \qquad \text{(d)}$$

$$Y_{ij} = 0, 1 \qquad \forall j \qquad \text{(e)}$$

$$X_j = 0, 1 \qquad \forall i,j \qquad \text{(f)}$$

h_i: Amount of demand at demand node i.

d_{ij}: Distance between demand node i and site j.

Y_{ij}: 1 if demand node i assigned to facility j. Else 0.

X_j: 1 if a facility located at site j. Else 0.

Objective function minimizes the demand-weighted distance between demand nodes and facility. First constraint shows that all of demand at that demand node must be satisfied. Second constraint shows that exactly p facilities are located.

In this Project, we are assigned to solve two different problem which named as “eil51” and “eil76”. As we can understand from the names of these problems, in problem eil51, we have 51 demand nodes, 51 x and y coordinates. In problem eil76, we have 76 demand nodes, 76 x and y coordinates. Distance approximation will be done with Euclidean distance formula.

Program Structure

Our program has five main excel sheet. First one is prepare sheet. In prepare sheet, we enter the city numbers and p value. There are two buttons as “Apply” and “Check Validity & Start” when we press apply, it will construct a data entry table. When data is entered, you may press check validity & start button it will transfer demand and coordinate data to model sheet. In model sheet, we have two tables. First one includes data table and second one includes demand-weighted distance values. When we press start button, code will show us objective values for each location and which facility is cheaper to go from each city. Besides that, it will show the facilities that opened under “IsOpen” title Then, it will lead us to NextGen sheet. In NextGen sheet, we can clearly see objective function values for all generations that code created with their generation number.

Code Explanation – Genetic Algorithm

The codes start by taking necessary parameters. These parameters are entry *number*, *iteration number*, *Generation Size* and *p-size. Entry Number* represent how many times the code will start all over again and create a randomly determined first Generation. *Iteration Number* stands for how many time the code will create new generation. *Generation Size* will determine how many creature will be created in each generation. If this number is too few than it will create a problem since the population will not be able sustain itself. *P* value is given parameter which determines how many True value will there be in an array of total number of cities.

When starting a generation, a randomly created Generation will appear. This generation will be composed of individuals or creatures which all the creatures are a feasible solution. An individual will be constructed by assigning a random number to each of its node. If that value is smaller than the value of p/*size* it will be have a True value. *p*/*size* is because in the long run we expect p number of nodes to be true for *size* many nodes. Generating a random number for each node might seem absurd but this is to conserve the uniformity of the individual. Distortion of uniformity is caused by the random number that is drawn using the programming language. With the way, we do it this uniformity is relatively conserved. We will check every creature's feasibility and if the creature is not feasible we will make an abortion. This process will be repeated for every individual in a generation. At the end, first initial generation will be created. This first generation will be presented in a sorted way.

After this part, iterative part will start with breeding new individuals and by killing not fit ones. An individual will be unfit by these conditions.

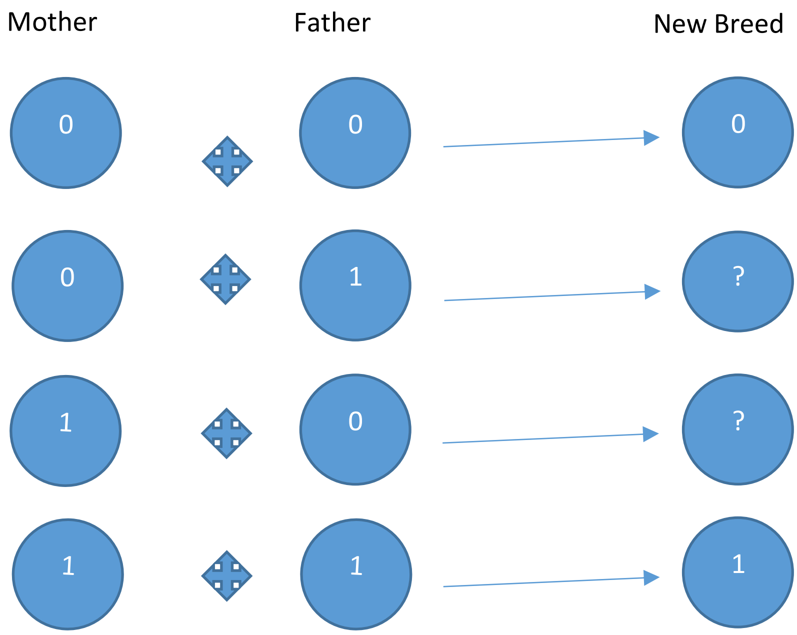
fitness(bestfit) + generationSpan * 0.2 < fitness(i)

60% of the individuals will die randomly

fitness(bestfit) + generationSpan * 0.7 < fitness(i)

all of them will die.

We will place a new breed to every dead once place. By using this breading algorithm.



When new breed is created, it is first tested if it is a feasible individual. If not we make an abortion. If feasible then we control if there is a mutation. There are four types of mutation category 0,1,2 and 3. Type of mutation is also the number open nodes will be closed and number of closed nodes that will be opened. If the child survives all this it will be place if no same configuration is already existing in the next generation. This control sort of acts like a taboo list. It will also prevent the convergence of the generation to a single individual.

Best answer and its configuration will be recorded to Memory Sheet it will also be recorded to Model in addition to the best solutions at every generation. Paragraph 2,3,4 & 5 will be repeated by the Entry Number times.

Solutions

We ran the code three times for all p values and wrote solutions below.

Problem eil51:

P = 3 => 8822 / 8859,183 / 9084

P = 4 => 7721,65 / 7789,5 / 7850,69

P = 5 => 6869,279 / 6972,195 / 6965,57

Facility Locations:

For p=3

1st trial (17,48,50)	2nd trial (16,17,48)	3rd trial (37,48,50)

For p=4

1st trial (17, 20, 48, 49)	2nd trial (17, 22, 23, 50)	3rd trial (17, 22, 48, 50)

For p=5

1st trial (19, 20, 26, 47, 49)	2nd trial (22, 26, 42, 47, 49)	3rd trial (19,22, 26, 47, 49)

Problem eil76:

P = 9 => 14 431 / 14 451,48 / 14 582,55

P = 10 => 12 892,36 / 13 759,29 / 13 633,38

P = 11 => 12 614,45 / 12 714,67 / 12 887,54

Facility Locations:

For p=9

1st trial (1, 19,21, 37, 39, 41, 44, 65, 76) =14 431

2nd trial (11, 19, 21, 32, 37, 41, 67, 72, 73) = 14 451,48

3rd trial (19, 27, 32, 36, 41, 65, 72, 73, 76) = 14 582,55

For p=10

1st trial (13, 21, 37, 38, 39, 41, 44,53, 59, 63) = 12 892,36

2nd trial (19, 27, 28, 32, 37, 38, 42, 63, 72, 75) = 13 759,29

3rd trial (13, 14, 21, 23, 32, 36, 38, 42, 73, 76) = 13 633,38

For p=11

1st trial (1, 3, 18, 19, 21, 27, 37, 38, 39, 41, 76) = 12 614,45

2nd trial (1, 3, 18, 19, 21, 27, 37, 38, 39, 41, 76) = 12 714,67

3rd trial (1, 3, 18, 19, 21, 27, 37, 38, 39, 41, 76) = 12 887,54