Princeton University
Department of Operations Research
and Financial Engineering

ORF 201
Computer Methods in Problem Solving

Lab 6: Facility Location

Due Sunday, Mar 26, 11:59 pm

---

1. INTRODUCTION

This assignment is about finding the best location for a "facility". We assume that there are n "customers" that already exist and are located at specific places. There are many different real-world applications for problems of this kind including

- Centrally locating a factory relative to existing distribution warehouses.
- Locating a central switching station in a communication network.
- Centrally locating a state capitol relative to the other populous cities in the state.
- Circuit placement in a computer chip.
- Locating a hub airport for an airline that uses hub-and-spoke routing.

The facility needs to be placed in a location that minimizes the facility-to-customer travel distances. While specific real-world situations may introduce various complications, such as computing distances along an existing road network, we shall simply assume that the objective is to minimize the sum of the straight-line distances from the facility to each of the n given customers.

One obvious suggestion would be to place the facility at the *centroid* of the n customers. Denoting the location vector of the i-th customer by $\mathbf{p}_i = (p_i^x, p_i^y)$, the centroid $\bar{\mathbf{p}}$ is very easy to calculate:

$$\bar{\mathbf{p}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{p}_i.$$

But this location does not minimize the sum of the distances. For example, consider 3 points on a line, say the x-axis. Assume that one of them is at position 0, another is at position 2, and the third is at position 100. The centroid is at position $(0+2+100)/3 = 34$. From this location, the sum of the distances is $34 + 32 + 66 = 132$. Now, consider placing the facility at position 2. The sum of the distances from this location is $2 + 0 + 98 = 100$. Hence, position 2 is much better than position 34. In fact, one can show that position 2 minimizes the sum of the distances.

It turns out that the centroid minimizes the sum of the *squares* of the distances. It is hard to imagine a logistics problem in which travel time or travel cost would be proportional to the square of the distance. Hence, the centroid is usually inappropriate in facility-location type problems.

The fact that the centroid minimizes the sum of the squares of the distances implies that customer locations that are far from the majority of the customers have too much influence over the centroid. For this reason, the centroid is said to put too much weight on *outliers*.

2. Weiszfeld's Iteration Scheme

Unfortunately, there is no simple formula for the location that minimizes the sum of the distances. There is, however, a simple iteration scheme that starts with an arbitrary location and iteratively computes better locations. If one were to run the iteration scheme for ever, it would converge to the location that minimizes the sum of the distances. Letting $\mathbf{q}_0 = (q_0^x, q_0^y)$ denote the (arbitrary) initial location, the iteration scheme works as follows:

$$\begin{aligned}
\mathbf{q}_1 &= \frac{\sum_i \mathbf{p}_i/\|\mathbf{p}_i - \mathbf{q}_0\|}{\sum_i 1/\|\mathbf{p}_i - \mathbf{q}_0\|} \\
\mathbf{q}_2 &= \frac{\sum_i \mathbf{p}_i/\|\mathbf{p}_i - \mathbf{q}_1\|}{\sum_i 1/\|\mathbf{p}_i - \mathbf{q}_1\|} \\
\mathbf{q}_3 &= \frac{\sum_i \mathbf{p}_i/\|\mathbf{p}_i - \mathbf{q}_2\|}{\sum_i 1/\|\mathbf{p}_i - \mathbf{q}_2\|} \\
&\vdots \\
\mathbf{q}_{k+1} &= \frac{\sum_i \mathbf{p}_i/\|\mathbf{p}_i - \mathbf{q}_k\|}{\sum_i 1/\|\mathbf{p}_i - \mathbf{q}_k\|} \\
&\vdots
\end{aligned}$$

This algorithm is called *Weiszfeld's iteration scheme*. Don't forget that boldface letters denote vectors and so, for example,

$$\|\mathbf{p}_i - \mathbf{q}_0\| = \sqrt{(p_i^x - q_0^x)^2 + (p_i^y - q_0^y)^2}.$$

It can be shown that for one-dimensional problems, the median of the locations minimizes the sum of the distances. Hence, the point that minimizes the sum of the distances in higher dimensions is called the *multidimensional median* and this assignment can be viewed as writing code to compute mulitdimensional medians.

3. Getting Started

As usual, you need to create some directories and copy some files. Begin like this:

```
    cd public_html/JAVA/ORF201
    mkdir median
    cd median
```

Then copy the following files:

```
    cp /u/orf201/public_html/JAVA/ORF201/median/index.html .
    cp /u/orf201/public_html/JAVA/ORF201/median/median.html .
    cp /u/orf201/public_html/JAVA/ORF201/median/Median.java .
```

4. Compiling

First, compile the java code that you just copied over:

```
    javac Median.java
```

Then use appletviewer to see what the code currently does:

```
    appletviewer median.html
```

An applet window should pop up with an editable text field for entering the number of customers, a button (labeled Mediate), and an empty drawing canvas below. At the moment, if you push the `Mediate` button nothing happens. That is because the main part of the code has been stripped out. It is your job to fill it in according to the instructions given below.

5. Programming Notes

In the method that computes each iteration of the Weiszfeld scheme, you'll need to put a call to `paint()`:

```
    paint(getGraphics());
```

If you want to slow down the calculation so that you have time to watch the animation, after the call to `paint()` you can add the following line:

```
    try { Thread.sleep(150); } catch(InterruptedException ie){}
```

The number 150 in the call to `Thread.sleep()` instructs the program to sleep for 0.150 seconds before continuing. Of course, the value 150 can be adjusted as you feel appropriate.

6. Median.html

Don't forget to edit `median.html` to make the following changes:

- Change `codebase` from `"../.."` to `http://www.princeton.edu/∼yourname/JAVA`.
- Add the honor code pledge:

  This program represents my own work in accordance with University regulations.

7. Going Public

If your umask is set for restricted access (i.e., 077), don't forget to make your files public:

```
chmod a+rx public_html/JAVA/ORF201/median
chmod a+r public_html/JAVA/ORF201/median/index.html .
chmod a+r public_html/JAVA/ORF201/median/median.html .
chmod a+r public_html/JAVA/ORF201/median/Median.class .
```

After you've made your files public check to see if you can bring your applet up in a browser. Fire up netscape and go to the following address:

```
http://www.princeton.edu/˜yourname/JAVA/ORF201/median/median.html
```

If your code works and permissions are set correctly, you should see the median applet in the browser.

8. Minimum Requirements

At a minimum:

(1) Implement Weiszfeld's iteration scheme.
(2) Develop and implement a reasonable stopping rule for the iteration scheme.
(3) Depict the successive iterations graphically. Draw lines connecting the current approximate multidimensional median to each of the customer locations.
(4) Compute and plot the centroid.
(5) Compute and plot the location whose x-component is the one-dimensional median of the x-components of each customer location and whose y-component is the one-dimensional median of the y-components of each customer location.
(6) Define methods as appropriate to partition the problem into well-defined subtasks.