Competititve location strategies for two facilities

Stephan Keller, Institute for Theoretical Computer Science

24. Januar 2018

Content

Competititve location strategies for two facilities

Zvi Drezner (Michigan University) published the paper[1] in 1981.

Real world application

vs.

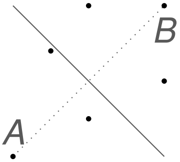
Problem 1

Theory

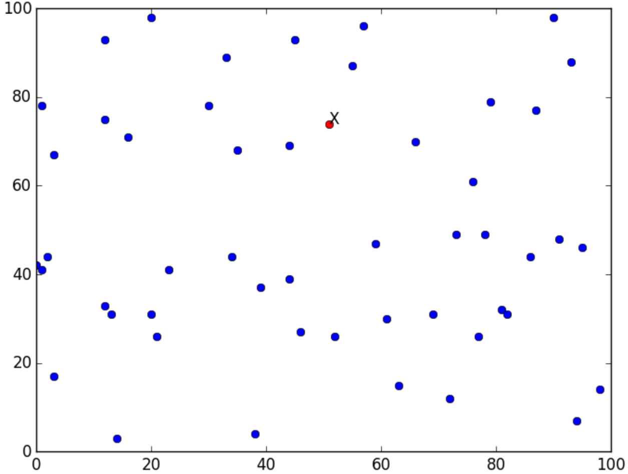
Problem 1

Definition

Given a location of an existing facility X serving the demand points, find a location for a new facility Y that will attract the most buying power of demand points.



Perpendicular bisector


100
80
60
40
20
0
0
20
40
60
80
100
X

Theorem 1

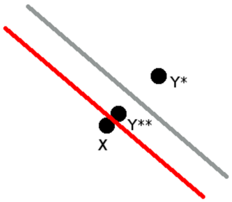
One of the optimal locations for Y when X is given is infinitesimally close to X but not on X.

Theorem 1

One of the optimal locations for Y when X is given is infinitesimally close to X but not on X.

Proof 1

When Y is located with X the total buying power at Y is zero because the distance between a customer and Y is equal to the distance between the customer and X, and therefore all customers will buy at X. Therefore, no optimal solution to Y is at X. Let Y* different from X's location be an optimal location for Y. A site Y** on the open segment connecting Y* and X is at least as good as Y* because the Y-half-plane for Y** is larger than the Y-half-plane for Y*. Therefore, the theorem follows. QED



Problem 1

Implementation

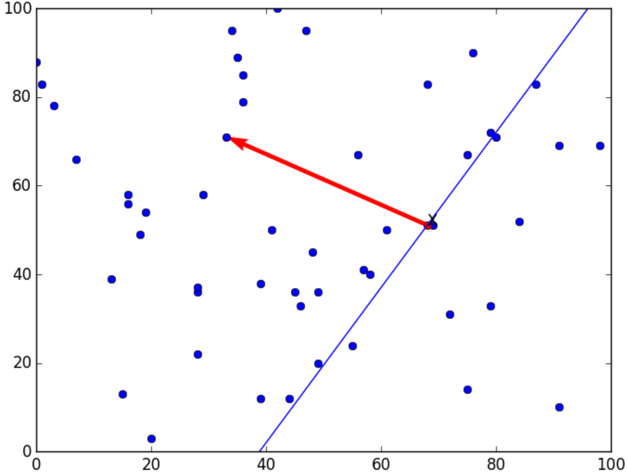
Algorithm

1. Find the direction of each demand Point from X sort these
2. Sort these directions in an increasing order between 0 and 2π
3. Calculate the buying power of all possible Y-halfplanes.
4. Determine the halfplane with the highest buying power

The complexity is $\mathcal{O}(n \log n)$ due to the sorting in 2, where n is the number of demand points.

Code[2]

```
def find_optimal_location_for_Y():
  point_directions =
      calculate_and_sort_all_point_directions()
  max_buying_power = 0
  best_half_plane_vector = [0,0]
  for angle, vector in point_directions:
    buying_power = get_points_in_halfplane(vector)
    if (buying_power>max_buying_power):
      max_buying_power=buying_power
      best_half_plane_vector = vector
```


100
80
60
40
20
0
0
20
40
60
80
100

Problem 2

Theory

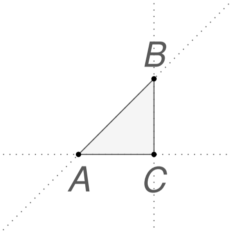
Problem 2

Definition - the Stackelberg equilibrium problem

Find a location for X such that it will retain the most buying power against a best possible location for an additional facility Y. The objective function to be minimized, for X is the buying power for a best possible location of Y.

Theorem 2

The intersection of all convex hulls for the sets with buying power of at least P_0 is identical with the intersection of all half-planes whose buying power is at least P_0.



Proof 2

Let Z be a point not in the intersection of the convex hulls.
Therefore: there must exist a polygon that Z does not belong to it and a side of the polygon such that Z is not in the half-plane defined by this side. Since the buying power of this half-plane is at least as the buying power of the set defining the polygon, the buying power of this half-plane is at least P_0 and therefore Z is not in the intersection of the half-planes. Conversely, if Z does not belong to the intersection of all half-planes with buying power of at least P_0 then there exists a half-plane that Z is outside it. This half-plane is defining a set of buying power of at least P_0, which Z is outside its convex hull. QED.

Problem 2

Implementation

P_i ... buying power for the i-th halfplane

1. Calculate all lines through pairs of points and calculate all P_i for each half-plane defined by the lines.

P_i ... buying power for the i-th halfplane

1. Calculate all lines through pairs of points and calculate all P_i for each half-plane defined by the lines.
2. Sort P_i in decreasing order. Set P_{min} and P_{max} to the lowest and highest P_i respectively.

P_i ... buying power for the i-th halfplane

1. Calculate all lines through pairs of points and calculate all P_i for each half-plane defined by the lines.
2. Sort P_i in decreasing order. Set P_{min} and P_{max} to the lowest and highest P_i respectively.
3. Set P_0 to the median value in the P_i vector for all $P_{min} < P_i < P_{max}$. If there is no P_i fulfilling $P_{min} < P_i < P_{max}$ go to step 7.

P_i ... buying power for the i-th halfplane

1. Calculate all lines through pairs of points and calculate all P_i for each half-plane defined by the lines.
2. Sort P_i in decreasing order. Set P_{min} and P_{max} to the lowest and highest P_i respectively.
3. Set P_0 to the median value in the P_i vector for all $P_{min} < P_i < P_{max}$. If there is no P_i fulfilling $P_{min} < P_i < P_{max}$ go to step 7.
4. Find if there is a feasible point to all half-planes for which $P_i \geq P_0$. This can be done by linear programming.

5. If there is a feasible solution point to the problem in step 4 then $min_x\{f(X)\} < P_0$. Set P_{max} to P_0, and go to step 3.

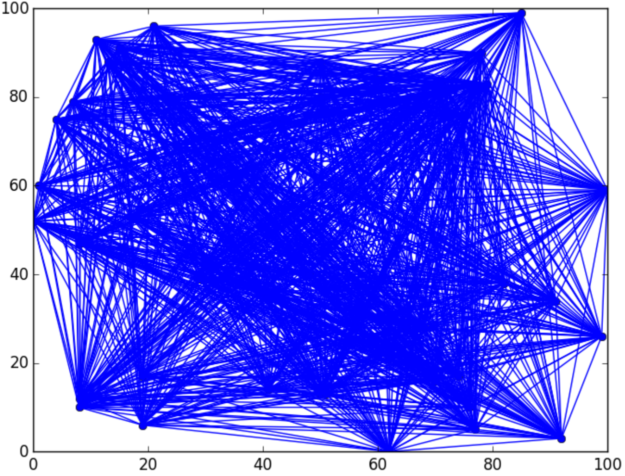
The algorithm is polynomial since the linear programming part of it is polynomial by Khachian[3].

5. If there is a feasible solution point to the problem in step 4 then $min_x\{f(X)\} < P_0$. Set P_{max} to P_0, and go to step 3.
6. Otherwise $min_x\{f(X)\} \geq P_0$. Set P_{min} to P_0, and go to step 3.

The algorithm is polynomial since the linear programming part of it is polynomial by Khachian[3].

5. If there is a feasible solution point to the problem in step 4 then $min_x\{f(X)\} < P_0$. Set P_{max} to P_0, and go to step 3.
6. Otherwise $min_x\{f(X)\} \geq P_0$. Set P_{min} to P_0, and go to step 3.
7. A feasible point for the last P_{max} is an optimal solution. The value of the objective function is P_{min}.

The algorithm is polynomial since the linear programming part of it is polynomial by Khachian[3].


0
20
40
60
80
100
0
20
40
60
80
100

Modified Problem

Modification

One may wish, in order to be more practical, to ask for a minimal distance requirement between Y and X when X is set up. In other words, we are not allowed to locate Y within a circle of a given radius $R \geq 0$ centered at X's location.

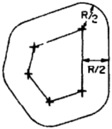


Fig. 2. The convex hull for the modified Problem 2.

Livedemo

References

[1] https://deepblue.lib.umich.edu/bitstream/handle/2027.42/23821/0000060.pdf;sequence=1

[2] https://github.com/stekeller/competitive.git

[3] https://www.sciencedirect.com/science/article/pii/0041555380900610