Métodos de Apoio à Decisão

Location problems: k-median, k-center, k-cover

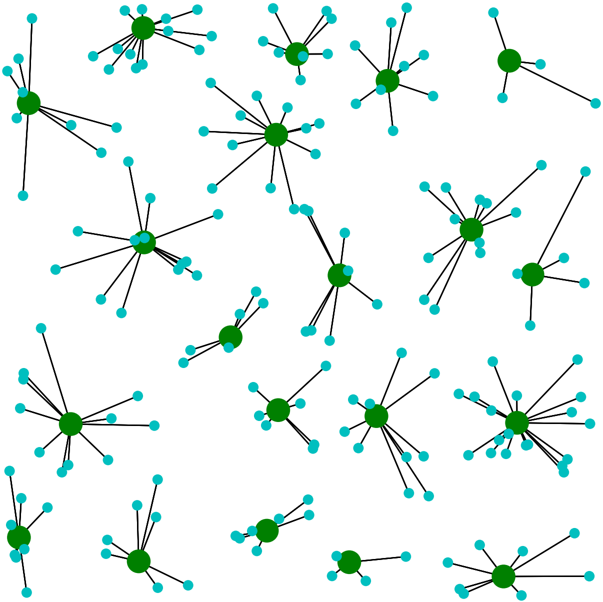
João Pedro Pedroso

2019/2020

Facility Location Problems

- k-Median
- k-Center
- k-Cover

The k-Median problem



The k-Median problem

Median problem

Select a given number of facilities from possible points in a graph, in such a way that the sum of the distances from each customer to the closest facility is minimized

- Often, the number k of facilities to be selected is predetermined in advance
- k median problem:
  - variant of uncapacitated facility location problem
  - seeks to establish k facilities without considering fixed costs
  - each demand point serviced by exactly one facility
  - objective: service all demand points at minimum total cost

The k-Median problem

- Notation:
  - distance from customer i to facility j $\rightarrow$ c_{ij}
  - set of customers $\rightarrow$ $\{1, l \ldots, n\}$
  - set of potential places for facilities $\rightarrow$ $\{1, \ldots, m\}$
  - commonly, facilities and customers share the same set of points
- Variables:

$$x_{ij} = \begin{cases} 1 & \text{when the demand of customer } i \text{ is met by facility } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{when facility } j \text{ is open} \\ 0 & \text{otherwise} \end{cases}$$

The k-Median problem

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \\
\text{subject to:} \quad & \sum_{j=1}^{m} x_{ij} = 1 \text{ for } i = 1, \cdots, n \\
& \sum_{j=1}^{m} y_j = k \\
& x_{ij} \le y_j \text{ for } i = 1, \cdots, n; j = 1, \cdots, m \\
& x_{ij} \in \{0, 1\} \text{ for } i = 1, \cdots, n; j = 1, \cdots, m \\
& y_j \in \{0, 1\} \text{ for } j = 1, \cdots, m
\end{aligned}
$$

- each customer i is assigned to exactly one facility j

$$\sum_{j=1}^{m} x_{ij} = 1 \text{ for } i = 1, \cdots, n$$

- exactly k facilities are established

$$\sum_{j=1}^{m} y_j = k$$

- force facility j to be open if it services demand point i

$$x_{ij} \leq y_j \text{ for } i = 1, \cdots, n; j = 1, \cdots, m$$

  - weaker formulation is obtained if we replace these nm constraints by n constraints

$$\sum_{i=1}^{n} x_{ij} \leq y_j, \text{ for } j = 1, \cdots, m$$

$\rightarrow$ lead to worse values in the linear relaxation

AMPL model

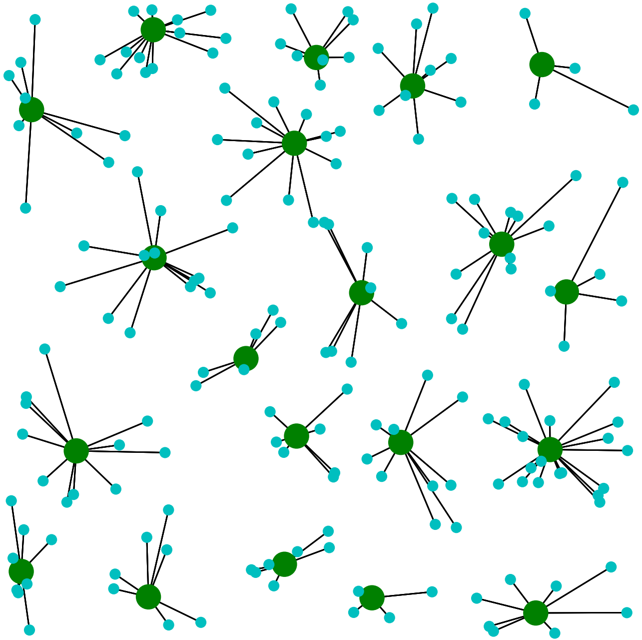
```
param n;   # number or customers
param m;   # number or facilities
param c {1..n, 1..m};
param k;

var x {1..n, 1..m} binary;
var y {1..m} binary;

minimize cost: sum {i in 1..n, j in 1..m} c[i,j] * x[i,j];

subject to
Service {i in 1..n}: sum {j in 1..m} x[i,j] = 1;
Kfacil: sum {j in 1..m} y[j] = k;
Activate {i in 1..n, j in 1..m}: x[i,j] <= y[j];
```

AMPL data

```
data;
param n := 5;
param m := 3;
param k := 2;
param c (tr) :    # (tr) --> transposed
            1  2  3  4  5 :=
      1     4  5  6  8  10
      2     6  4  3  5  8
      3     9  7  4  3  4 ;
```

Illustration

- Solution obtained for a graph with 200 vertices placed randomly in the two-dimensional unit box
- Costs given by Euclidean distance
- Each of the vertices is a potential location for a facility

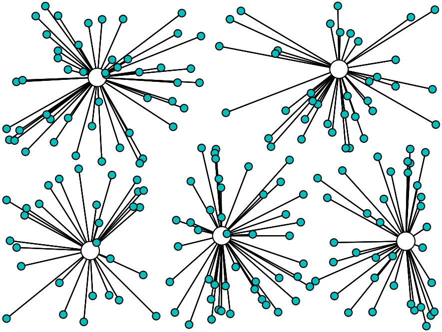


Programming

- AMPL model: see above
- AMPL data: replaced by a Python program

$\rightarrow$ see file `kmedian_plt.py`

The k-Center Problem



The k-Center Problem

Center problem

Select a given number of facilities from possible points in a graph, in such a way that the maximum value of a distance from a customer to the closest facility is minimized.

- Variant of the k-median problem
- Assign facilities to a subset of vertices
  - aim: each customer "close" to some facility
- Number k of facilities is predetermined

The k-Center problem

- Notation (same as k-Median):
  - distance from customer i to facility $\rightarrow j\ c_{ij}$
  - set of customers $\rightarrow \{1, \ldots, n\}$
  - set of potential places for facilities $\rightarrow \{1, \ldots, m\}$
  - commonly, facilities and customers share the same set of points
- Variables (same as k-Median):

$$x_{ij} = \begin{cases} 1 & \text{when the demand of customer } i \text{ is met by facility } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{when facility } j \text{ is open} \\ 0 & \text{otherwise} \end{cases}$$

- Distance/cost for most distant customer from an activated facility
  - additional continuous variable z

The k-Center problem

$$
\begin{aligned}
\text{minimize} \quad & z \\
\text{subject to:} \quad & \sum_{j=1}^{m} x_{ij} = 1 && \text{for } i = 1, \cdots, n \\
& \sum_{j=1}^{m} y_j = k \\
& x_{ij} \leq y_j && \text{for } i = 1, \cdots, n; j = 1, \cdots, m \\
& \sum_{j=1}^{m} c_{ij} x_{ij} \leq z && \text{for } i = 1, \cdots, n \\
& x_{ij} \in \{0, 1\} && \text{for } i = 1, \cdots, n; j = 1, \cdots, m \\
& y_j \in \{0, 1\} && \text{for } j = 1, \cdots, m
\end{aligned}
$$

- New constraint:

$$\sum_{j=1}^{m} c_{ij} x_{ij} \leq z \quad \text{for } i = 1, \cdots, n$$

- Determine z to take on at least c_{ij}
  - for all facilities j and customers i assigned to j
  - weaker (maybe more natural) version:

$$c_{ij} x_{ij} \leq z, \text{ for } i = 1, \cdots, n; j = 1, \cdots, m$$

  - intuition: in the strong formulation we are adding more terms in the left-hand side $\rightarrow$ feasible region is tighter
- New objective: z
  - minimizing a maximum value $\rightarrow$ min-max objective
  - type of problems for which mathematical optimization solvers are typically weak
  - instead of the previous objective

$$\text{minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

AMPL model

```
param n;   # number or customers
param m;   # number or facilities
param c {1..n, 1..m};
param k;

var x {1..n, 1..m} binary;
var y {1..m} binary;
var z >= 0;

minimize maxcost: z;

subject to
Service {i in 1..n}: sum {j in 1..m} x[i,j] = 1;
Kfacil: sum {j in 1..m} y[j] = k;
Activate {i in 1..n, j in 1..m}: x[i,j] <= y[j];
MinZ {i in 1..n}: sum {j in 1..m} c[i,j] * x[i,j] <= z;
```

Programming: same as with kmedian

- Observe difference in performance
- Observe difference in the solution

Techniques in linear optimization

- "Minimization of the maximum value" can be reduced to a standard linear optimization
  - add new variable
  - make that variable at least as large as each of the values
- Assume that we want to minimize the maximum of two linear expressions:
  - $3x_1 + 4x_2$
  - $2x_1 + 7x_2$.
  - minimize new variable z subject to:

$$3x_1 + 4x_2 \leq z$$
$$2x_1 + 7x_2 \leq z$$

Related topic: minimization of the absolute value

Minimization of the absolute value $|x|$ of a real variable x:

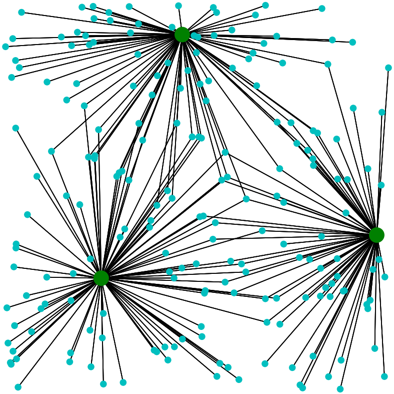
- Nonlinear expression
- To linearaize it: add non-negative variables y and z
  - $x = y - z \rightarrow$ value of x in terms of y and z
  - now, $|x|$ can be written as $y + z$
- So:
  - occurrences of x in the formulation $\rightarrow$ replaced by $y - z$
  - $|x|$ in the objective function $\rightarrow$ replaced by $y + z$.
- Another possibility:
  - adding variable z
  - impose $z \geq x$ and $z \geq -x$
  - z replaces $|x|$ in the objective function

Modeling tip

An objective function that minimizes a maximum value should be avoided, if possible.

- In integer optimization $\rightarrow$ solved by the branch-and-bound method
- If the objective function minimizes the maximum value of a set of variables:
  - tendency to have large values for the difference between the lower bound and the upper bound (the so-called duality gap).
  - time for solving the problem becomes large
  - if branch-and-bound is interrupted, the incumbent solution is rather poor.

The k-Cover Problem



The k-Cover Problem

- Variant to k-center problem
- Avoids the min-max objective
- Process makes use of binary search

The k-Cover Problem

- Graph $G_\theta = (V, E_\theta)$
  - set of edges whose distances from a customer to a facility which do not exceed a threshold value θ
  - edges: $E_\theta = \{\{i, j\} \in E : c_{ij} \leq \theta\}$
- Subset $S \subseteq V$ is called a cover if every vertex $i \in V$ is adjacent to at least one of the vertices in S
- Idea: optimum value of the k-center problem $\leq \theta$ if there exists a cover with cardinality $|S| = k$ on graph G_θ.

Notation

- Variables:
  - $y_j = 1$ if a facility is opened at j, 0 otherwise
    - vertex j is in the subset S or not
  - $z_i = 1$ if vertex i is adjacent to no vertex in S, 0 otherwise
    - vertex i is covered or not
- $[a_{ij}] \rightarrow$ incidence matrix of G_θ
  - $a_{ij} = 1$ if vertices i and j are adjacent, 0 otherwise
- We need to determine whether or not graph G_θ has a cover $|S| = k$
- We can do that by solving integer-optimization model $\rightarrow$ k-cover problem on G_θ

The k-Cover Problem: model

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} z_i \\
\text{subject to} \quad & \sum_{j=1}^{m} a_{ij} y_j + z_i \geq 1 && \text{for } i = 1, \cdots, n \\
& \sum_{j=1}^{m} y_j = k \\
& z_i \in \{0, 1\} && \text{for } i = 1, \cdots, n \\
& y_j \in \{0, 1\} && \text{for } j = 1, \cdots, m.
\end{aligned}
$$

- adjacency matrix is built upon a given value of distance θ
  - used to compute set of facilities that may service each of the customers within that distance

Binary search

- Given θ, either:
  - optimal objective value of the previous optimization problem is zero
    - k facilities were enough for covering all the customers withing distance θ
    - $\rightarrow$ reduce θ
  - greater that zero
    - there is at least one $z_i > 0$
    - thus, a customer could not be serviced from any of the k open facilities
    - $\rightarrow$ increase θ
- binary search: repeate until bounds for θ are close enough

AMPL model

```
param n;   # number or customers
param m;   # number or facilities
param c {1..n, 1..m};
param k;
param theta;

var y {1..m} binary;
var z {1..n} binary;

minimize cost: sum {i in 1..n} z[i];

subject to
Service {i in 1..n}:
    sum {j in 1..m : c[i,j] <= theta} y[j] + z[i] >= 1;
Kfacil: sum {j in 1..m} y[j] = k;
```

Programming: loading the model

```
random.seed(1)
n = 200
m = n
I,J,c,x_pos,y_pos = make_data(n,m)
k = 3

ampl = AMPL()
ampl.option['solver'] = 'gurobi'
ampl.read("kcover.mod")
ampl.param['n'] = n
ampl.param['m'] = n
ampl.param['k'] = k
ampl.param['c'] = c

print("solving")
start = time.time()
facilities,edges = [],[]
delta = 1.e-4  # tolerance
LB = 0
UB = max(c[i,j] for (i,j) in c)
```

Programming: binary search

```
delta = 1.e-4  # tolerance
LB = 0
UB = max(c[i,j] for (i,j) in c)
while UB-LB > delta:
    theta = (UB+LB) / 2.
    ampl.param['theta'] = theta
    ampl.solve()
    cost = ampl.pbj['cost']
    if cost.value() < delta:
        UB = theta
        y_ = ampl.var['y']
        facilities = [j for j in J if y_[j].value() > .5]
        edges = [(i,j) for i in I for j in facilities if c[i,j] < theta]
    else:   # infeasibility > 0:
        LB = theta
```

Remarks

- k-cover within binary search $\rightarrow$ time comparable k-median check
- in practice, k-center solution is usually preferable to k-median
  - longest time required for servicing a customer
  - may be large on the k-median solution.