Optimization problems in logistics: location

Logistics

Giovanni Righini

Università degli Studi di Milano

Location theory

Location theory is a branch of Operations research and Mathematical programming in particular.

It is concerned with algorithms to compute optimal locations for resources, service centers, and so on.

It obviously applies to logistics but not only to logistics: it has applications in many contexts in which locations have to be determined in graphs (e.g. optimally locating ambulances in a city) or in Euclidean spaces (e.g. optimally locating antennas in a territory).

Decisions

In logistics, locational decisions are typically taken at a strategic level or less frequently at a tactical level.

Costs usually take into account

- cost terms that depend only on the location chosen (e.g. building, renting, maintaining a warehouse);
- cost terms that depend on the interaction with the demand (e.g. transportation costs, distances between service centers and customers' locations,...).

Locating facilities induces a (geographical) partition of the demand: location/allocation problems.

Classification

Location problems can be classified according to several criteria.

- Decision space:
  - continuous space (usually $\Re^2$) (continuous location);
  - graphs:
    - location restricted to the vertices/nodes (discrete location);
    - location allowed also on the edges/arcs (continuous location).
- Number/type of facilities: single-facility, multi-facility,...
- Objective: min-sum, min-max, max-cover,...
- Constraints: capacity, forbidden regions,...
- Time horizon: single-period or multi-period.
- Type of facility: single-type or multi-type.
- Type of material flow: single-commodity or multi-commodity.
- Interaction among facilities (e.g. hub airports).
- Flow direction: inbound, outbound, both.
- Splittable demand.
- Competitive location.
- Obnoxious location.
- Hybrid models: location/inventory, location/routing, location/scheduling, etc...

Objectives

Median of a set of points N: a point X that minimizes the sum of the distances between X and the points in N (min-sum).

$$\text{minimize } z(X) = \sum_{P_i \in N} dist(X, P_i).$$

Center of a set of points N: a point x that minimizes the maximum of the distances between X and the points in N (min-max).

$$\text{minimize } z(X) = \max_{P_i \in N} dist(X, P_i).$$

Weighted medians and centers minimize $\sum_{P_i \in N} w_i dist(X, P_i)$ and $\max_{P_i \in N} w_i dist(X, P_i)$, where w_i is a given weight for each point $P_i \in N$.

Location/allocation

When a given number of facilities must be located, it is indicated by p:

- min-sum: p-median problem;
- min-max: p-center problem.

The set N is partitioned into p subsets, one assigned to each facility (obviously, the closest one).

A problem has the single-source constraint if each point $P_i \in N$ must be assigned to a unique facility.

A problem has the single-assignment property if it has at least an optimal solution complying with the single-source constraint, although the contraint is not explicitly enforced.

In their basic version, uncapacitated location problems have the single-assignment property.

Distances

In continuous spaces, distances can be defined as

- Euclidean distances: $dist(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ (L^2 norm)
- Manhattan distances: $dist(P_i, P_j) = |x_i - x_j| + |y_i - y_j|$ (L^1 norm)
- Others. For instance: $dist(P_i, P_j) = \max\{|x_i - x_j|, |y_i - y_j|\}$ (L^∞ norm)

In graphs, distances are defined as shortest path lengths.

In both cases:

- $dist(P_i, P_i) = 0$
- $dist(P_i, P_j) > 0 \ \forall P_i \neq P_j$
- $dist(P_i, P_j) = dist(P_j, P_i)$
- $dist(P_i, P_j) \leq dist(P_i, P_k) + dist(P_k, P_j) \ \forall P_i, P_j, P_k$

In some cases some of these properties may not hold (e.g. digraphs).

Location in $\Re^2$

Location problems in continuous spaces typically involve Euclidean distances, which are non-linear in the variables:

$$dist(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Typical examples are:

- finding the 1-median in $\Re^2$: the Weber problem (easy: Weiszfeld algorithm)
- finding the p-medians in $\Re^2$: the multi-Weber problem (difficult: Cooper heuristic)
- finding the 1-center in $\Re^2$: the Sylvester problem (easy: Elzinga-Hearn algorithm)
- finding the p-centers in $\Re^2$: the p-center problem (difficult: heuristics).

1-median in $\Re^2$: Weiszfeld algorithm

- Initialize a point x.
- For each point not coinciding with x compute the first derivative of its distance in x.
- Compose all vectors and compute x'.
- If last step is small then stop, else repeat.

The problem is convex: there is a unique global minimum.

The optimal solution is better and better approximated, not necessarily reached exactly.

1-center in $\Re^2$

One of the shortest papers ever (one line and a half long):

J.J.Sylvester, *A question in the geometry of situation*, Quarterly Journal of Pure and Applied Mathematics 1, 79 (1857).

“It is desired to find the least circle which shall contain a given system of points in a plane.”

1-center in $\Re^2$: Elzinga-Hearn algorithm

- Select a subset $\mathcal{P}$ of three of the given points, at random.
- Find the center x of the smallest circle $\mathcal{C}$ covering them.
- Find a given point $P' \notin \mathcal{P}$ farthest from x.
- Find the center x of the smallest circle $\mathcal{C}'$ covering the points in $\mathcal{P} \cup \{P'\}$.
- Delete from $\mathcal{P}$ the points stricly covered by $\mathcal{C}'$.
- Repeat until all points are covered.

The theoretical complexity is quadratic, but in practice the running time grows linearly with the number of given points.

p-medians: Cooper algorithm

- Initialize p medians $x_1, x_2, \ldots x_p$ in some way.
- For each given point $j = 1, \ldots, n$ assign it to the current closest median.
- For each cluster of points assigned to the same median, compute the optimal location of the median.
- If nothing has changed, stop; else repeat.

This algorithm is very easy to code, but it can produce arbitrarily bad solutions.

Medians

Node optimality property. Optimal location of (weighted) medians always occurs in the nodes/vertices of the graph.

Proof. The proof uses the following lemmata.

1. A linear function is concave.
2. If $f(x)$ is concave and $\lambda \geq 0$, then $\lambda f(x)$ is concave.
3. If $f_1(x)$ and $f_2(x)$ are concave functions, then $f_1(x) + f_2(x)$ is also concave.
4. If $f(v)$ is a concave function of vector v and the components of v are linear in x, then $f(v(x))$ is concave.
5. If $f_i(x)$ $i = 1, \ldots, n$ is a set of concave functions, then $\min_{i=1,\ldots,n}\{f_i(x)\}$ is concave.
6. If $f(x)$ is concave and defined in $[0, 1]$, then $\min_{0\leq x\leq 1}\{f(x)\} = \min\{f(0), f(1)\}$.

Then, when transportation costs are concave functions, the cost function on each arc is concave. Therefore, there exists at least a subset of p vertices that are optimal p-medians of the graph.

The 1-median problem

Given a weighted graph $G = (V, E)$, being d_{ij} the distance between any two vertices i and j, find the 1-median of the graph.

Binary variables: $x_i = 1$ if and only if the median is in vertex $i \in V$.

$$\begin{aligned}
\text{minimize } z = \sum_{i \in V} \sum_{j \in V} d_{ij} x_i & \\
\text{s.t. } \sum_{i \in V} x_i = 1 & \\
x_i \in \{0, 1\} & \qquad \forall i \in V.
\end{aligned}$$

The problem is easily solvable by enumeration.
All pairs shortest paths can be computed in $O(nm)$ time.

The 1-median on a tree

Given a Tree $T = (V, E)$, consider a generic edge $[v1, v2] \in E$ and the two trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ obtained from T by removing it, such that $v_1 \in V_1$ and $v_2 \in V_2$.

Let $x \in V$ be the 1-median of T.

Lemma 1. $x \in T_1$ if and only if $w(V_1) \geq w(V_2)$.

Lemma 2. If $w(V_1) \geq w(V_2)$, then the 1-median of T is the 1-median of T_1, where $w(v_1)$ is increased by $w(V_2)$.

Majority algorithm (Goldman, 1971).

1. Take any leaf v of T: if $w(v) \geq w(V)/2$, stop: v is the median.
2. Take the unique vertex u adjacent to v; increase $w(u)$ by $w(v)$; delete v from T. Go to 1.

Worst-case time complexity: $O(m)$.

The *p*-medians problem

Given a weighted graph $G = (V, E)$, being d_{ij} the distance between any two vertices i and j, and given an integer $1 < p < |V|$, select p medians of the graph.

The cost incurred by each vertex is the distance to its closest median. Hence the selection of the medians induces a partition of the graph into p vertex subsets.

Algorithms:

- locate-first-allocate-second;
- allocate-first-locate-second;
- iterated location-allocation.

The p-medians problem on a graph

Location variables: y_i binary, representing whether $i \in V$ is a median.
Allocation variables: x_{ij} binary, indicating whether $j \in V$ is assigned to the median in $i \in V$.

$$
\begin{aligned}
\text{minimize } z = \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} & \\
\text{s.t. } \sum_{i \in V} y_i = p & \\
\sum_{i \in V} x_{ij} = 1 & \qquad \forall j \in V \\
x_{ij} \leq y_i & \qquad \forall i \in V, \forall j \in V \\
y_i \in \{0,1\} & \qquad \forall i \in V \\
x_{ij} \in \{0,1\} & \qquad \forall i \in V \forall j \in V.
\end{aligned}
$$

The p-median problem is NP-hard (Kariv and Hakimi, 1979).
The integrality restrictions $x_{ij} \in \{0,1\}$ are redundant.
Variables y_i can be replaced by binary variables x_{ii}.

Algorithms

Enumeration algorithms require to enumerate and evaluate all vertex subsets of cardinality p.

They are viable only for small values of p (n. of medians) or n (n. of vertices).

The linear continuous relaxation provides integer solutions when

- the graph is a path;
- the graph has a single cycle;
- the graph is made by the above components.

For the general case the most effective algorithms are based on Lagrangean relaxation or column generation.

Graph-theoretic algorithms

Graph-theoretic algorithms are used when the underlying graph has a special structure; in particular, when it is a tree.

2**-medians on a tree.**

1. For each $e \in E$, consider the two trees T_1 and T_2 obtained from T by deleting e.
2. Compute the optimal 1-median for T_1 and T_2 separately; sum their costs.
3. Output the pair of optimal 1-medians of the (T_1, T_2) pair of minimum total cost.

Complexity: $O(n^2)$.

For general p, Kariv and Hakimi (1979) gave a D.P. algorithm with complexity $O(p^2n^2)$.

Centers

The node optimality property does not hold for optimal (weighted) centers.

Lemma 3 used for medians:
If $f_1(x)$ and $f_2(x)$ are concave functions, then $f_1(x) + f_2(x)$ is also concave.

If $f_1(x)$ and $f_2(x)$ are concave functions, then $\max\{f_1(x), f_2(x)\}$ is not necessarily concave.

Classification (Handler):

$$\{V,A\}/\{V,A\}/\{p,\lambda^{-1}\}/\{N,T\}$$

meaning

facility set / demand set / n.centers or max distance / type of graph

The restricted 1-center problem on a graph: $V/V/1/N$

Given a weighted graph $G = (V, E)$, being d_{ij} the distance between any two vertices i and j, find the 1-center of the graph.

Binary variables: $x_i = 1$ if and only if the center is in vertex $i \in V$.
Continuous variable: r, maximum distance between a vertex and the center.

$$\begin{aligned} \text{minimize } z = & r \\ \text{s.t.} \sum_{i \in N} x_i & = 1 \\ d_{ij} x_i & \leq r && \forall j \in V \\ x_i & \in \{0, 1\} && \forall i \in V. \end{aligned}$$

The problem is easily solvable by enumeration.

Absolute centers on a tree

Absolute 1-center problem on a tree: $A/V/1/T$ (Haendler, 1973).

1. Select any point x on the tree, arbitrarily.
2. Find the leaf u that is farthest from x.
3. Find the leaf v that is farthest from u.
4. Compute the midpoint of the path between u and v.

Complexity: $O(n)$.

Absolute 2-center problem on a tree: $A/V/2/T$ (Haendler, 1978).

1. Find the absolute 1-center x on the tree (midpoint between leaves u and v).
2. Delete any arc containing x and belonging to the path betweeen u and v.
3. Find the absolute 1-centers of the two resulting subtrees.

Complexity: $O(n)$.

Absolute p-centers problem on a tree: $A/V/p/T$
Complexity: $O(n \log n)$ (Frederickson and Johnson, 1983).

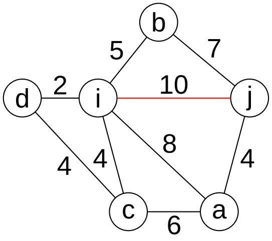
The absolute 1-center problem on a graph $A/V/1/N$

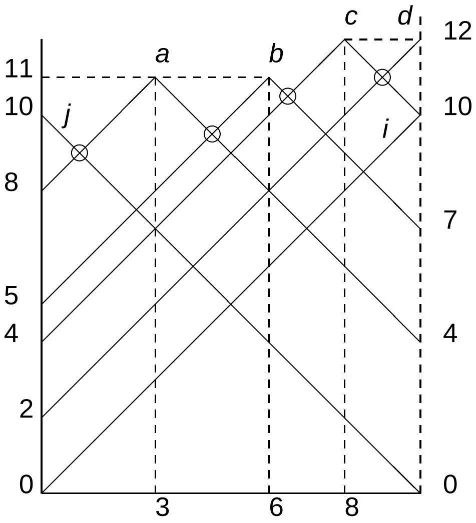
Given an undirected weighted graph $G = (V, E)$, for any two distinct vertices $u \in V$ and $v \in V$ and any edge $[i, j] \in E$, $x \in [i, j]$ is a *local center* if and only if

$$\begin{cases} dist(u, x) = dist(v, x) \\ i \in P(u, x) \\ j \in P(v, x) \end{cases}$$

There are at most n local centers on each edge.
Sorting them to compute their intersections takes $O(n \log n)$.
This must be repeated for each edge, i.e. m times.

Complexity: $O(mn \log n)$ (Kariv and Hakimi, 1979).


b
d
i
j
c
a
5
7
2
10
8
4
4
4
6


a
b
c
d
j
i
11
10
8
5
4
2
0
12
10
7
4
0
3
6
8

The absolute p-centers problem on a graph: $A/V/p/N$

Given a weighted graph $G = (V, E)$, being d_{ij} the distance between any two vertices i and j, and given an integer $1 < p < |V|$, select p centers of the graph.

The cost is the maximum distance between a vertex and the center of its cluster.
Hence the selection of the centers induces a partition of the graph into p vertex subsets.

All existing algorithms operate on the same principle: generating and solving a series of set covering problems.

Only local centers are candidate locations for the p centers.

Minieka algorithm (1970)

Let C be the set of local centers of the graph.

1. Compute a shortest path matrix F with a row for each vertex and a column for each local center in the graph.
2. Select an arbitrary subset of p columns, S.
3. Loop: Set $d := \max_{v \in V} \min_{x \in S} \{f_{vx}\}$
4. Set $b_{vx} := 0$ if $f_{vx} \geq d$ and $b_{vx} := 1$ otherwise.
5. Solve the set covering problem $\min z = \{e^T x : Bx \geq e, x \in \{0,1\}^{|C|}\}$.
6. If $z > p$, then stop: S is optimal. Otherwise a solution S' better than S has been obtained: go to Loop.

Minieka algorithm (improved version)

Let d_x the radius of the local center, i.e. its distance from the vertices defining it.

1. Compute a shortest path matrix F with a row for each vertex and a column for each local center in the graph.
2. Select an arbitrary subset of p columns, S.
3. Loop: Set $d := \max_{v \in V} \min_{x \in S} \{f_{vx}\}$
4. Set $b_{vx} := 0$ if $f_{vx} > d$ and $b_{vx} := 1$ otherwise.
5. Delete from matrix b all columns j s.t. $d_j \geq d$.
6. Solve the set covering problem $\min z = \{e^T x : Bx \geq e, x \in \{0,1\}^{|C|}\}$.
7. If $z > p$, then stop: S is optimal. Otherwise a solution S' better than S has been obtained: go to Loop.

Relaxation algorithm (Garfinkel, Neebe, Rao, 1977)

The most time consuming step of Minieka algorithm is the solution of the set covering problem.

In the *p*-center problem, most constraints are not binding at optimality.

The idea is to iteratively generate vertices to be inserted in the set covering problem instance.

These algorithms should be re-evaluated today, comparing them with general-purpose ILP solvers that were not available in the 70s.

The restricted p-centers problem on a graph: $V/V/p/N$

Location variables: $y_i = 1$ if and only if a center is in vertex $i \in V$.
Allocation variables: x_{ij} binary, indicating whether $j \in V$ is in the cluster with a center in $i \in V$.
Continuous variable: r, maximum distance between a vertex and its center.

$$
\begin{aligned}
\text{minimize } z = & r \\
\text{s.t. } & \sum_{i \in N} y_i = p \\
& \sum_{i \in V} x_{ij} = 1 && \forall j \in V \\
& x_{ij} \leq y_i && \forall i \in V, \forall j \in V \\
& d_{ij} x_{ij} \leq r && \forall i \in V, \forall j \in V \\
& y_i \in \{0,1\} && \forall i \in V \\
& x_{ij} \in \{0,1\} && \forall i \in V, \forall j \in V.
\end{aligned}
$$

The restricted p-centers problem on a graph: $V/V/p/N$

An algorithm by Toregas, Swan, ReVelle and Bergman (1971) resembles Minieka's algorithm for $A/V/p/N$.

The set of local centers C is replaced by the vertex set V.

The midpoint property no longer holds.

The relaxation algorithm can be extended to this case, but it does not provide substantial savings in computing time as it does with $A/V/p/N$.

Therefore $V/V/p/N$ can be even more difficult to solve than $A/V/p/N$.

Continuous demand: $A/A/p/N$ and $V/A/p/N$

$A/A/p/N$.

This is the most difficult case.

The relaxation approach can be extended to solve it.

Efficient algorithms are available for $A/A/p/T$ (Chandrasekaran and Tamir, 1980, Megiddo and Tamir, 1983).

$V/A/p/N$.

It is solvable by adapting the relaxation algorithm.

Inverse center problems

$A/V/\lambda^{-1}/N$.

Let z_λ be the minimum number of centers needed to cover the vertices within a radius λ.

A single set covering instance must be solved.

Extensions to $A/A/\lambda^{-1}/N$, $V/V/\lambda^{-1}/N$ and $V/A/\lambda^{-1}/N$ are straightforward.

Max covering location problem

Given a weighted graph $G = (V, E)$, given the distance d_{ij} between any two vertices i and j, given the demand q_i associated with each vertex $i \in V$, given an integer $1 < p < |V|$, given a radius r, select p vertices of the graph so that the maximum demand is covered.

A vertex is covered by a facility if and only if the distance between them is within the radius.

Max covering location problem

Location variables: $y_i = 1$ if and only if a facility is located in vertex $i \in V$.
Covering variables: $w_j = 1$ if and only if vertex $j \in V$ is covered.

$$
\begin{aligned}
\text{maximize } z = \sum_{j \in N} q_j w_j & \\
\text{s.t. } \sum_{i \in V} y_i = p & \\
w_j \leq \sum_{i \in V : d_{ij} \leq r} y_i & \qquad \forall j \in V \\
y_i \in \{0, 1\} & \qquad \forall i \in V \\
w_j \in \{0, 1\} & \qquad \forall j \in V.
\end{aligned}
$$

The problem is *NP*-hard, but it is solvable in polynomial time if the graph is a tree.

Single-echelon single-commodity location

Plant (capacitated) location problem: homogeneous locations, single-commodity, mono-directional flow (e.g. outbound only), linear or piece-wise linear cost for both transportation and operations at the facilities, splittable demand.

The problem is modelled with a complete *bipartite graph*:

- V_1 is the set of sites where a facility can be located;
- V_2 is the set of sites where customers are located;
- $A = V_1 \times V_2$ is the arc set connecting the two partitions.

Each potential facility has a capacity $q_i \; \forall i \in V_1$.
Each customer has a known demand $d_j \; \forall j \in V_2$.
The operational cost for each potential facility is $F_i(u_i) \; \forall i \in V_1$, where u_i indicates the total flow from i.
The transportation cost for each arc $(i, j) \in A$ is $C_{ij}(x_{ij})$, where x_{ij} indicates the amount of flow on the arc.

Single-echelon single-commodity location

The mathematical model is:

$$\begin{aligned}
\text{minimize } z = & \sum_{i\in V_1}\sum_{j\in V_2} C_{ij}(x_{ij}) + \sum_{i\in V_1} F_i(u_i) & \\
\text{s.t. } & \sum_{j\in V_2} x_{ij} = u_i & \forall i \in V_1 \\
& \sum_{i\in V_1} x_{ij} = d_j & \forall j \in V_2 \\
& 0 \le u_i \le q_i & \forall i \in V_1 \\
& x_{ij} \ge 0 & \forall i \in V_1, \forall j \in V_2
\end{aligned}$$

Additional constraints can be imposed such as capacities on the arcs or limits to the number of locations.

Fixed transportation price and facility set-up cost

Assume $C_{ij}(x_{ij}) = c_{ij}x_{ij}$ and $F_i(u_i) = \begin{cases} f_i & \text{if } u_i > 0 \\ 0 & \text{if } u_i = 0 \end{cases}$

To represent fixed costs we need to use a binary variable y_i replacing the continuous variable u_i for each potential facility $i \in V_1$.

$$\begin{aligned}
\text{minimize } z = & \sum_{i \in V_1} \sum_{j \in V_2} c_{ij} x_{ij} + \sum_{i \in V_1} f_i y_i \\
\text{s.t. } & \sum_{j \in V_2} x_{ij} \leq q_i y_i && \forall i \in V_1 \\
& \sum_{i \in V_1} x_{ij} = d_j && \forall j \in V_2 \\
& y_i \in \{0, 1\} && \forall i \in V_1 \\
& x_{ij} \geq 0 && \forall i \in V_1, \forall j \in V_2
\end{aligned}$$

The problem is formulated as a mixed-integer programming model.

Fixed and variable facility set-up costs

Assume that

$$F_i(u_i) = \begin{cases} f_i + g_i u_i & \text{if } u_i > 0 \\ 0 & \text{if } u_i = 0 \end{cases}$$

Since $u_i = \sum_{j \in V_2} x_{ij}$,

$$\begin{aligned}
\text{minimize } z = & \sum_{i \in V_1} \sum_{j \in V_2} (c_{ij} + g_i) x_{ij} + \sum_{i \in V_1} f_i y_i \\
\text{s.t. } & \sum_{j \in V_2} x_{ij} \le q_i y_i && \forall i \in V_1 \\
& \sum_{i \in V_1} x_{ij} = d_j && \forall j \in V_2 \\
& y_i \in \{0, 1\} && \forall i \in V_1 \\
& x_{ij} \ge 0 && \forall i \in V_1, \forall j \in V_2
\end{aligned}$$

Lower bounds on the activity level

Assume that a facility can profitably operate only if $u_i \geq m_i \; \forall i \in V_1$.
We can impose such a constraint in this way:

$$
\begin{aligned}
\text{minimize } z = \sum_{i \in V_1} \sum_{j \in V_2} (c_{ij} + g_i) x_{ij} + \sum_{i \in V_1} f_i y_i & \\
\text{s.t. } \sum_{j \in V_2} x_{ij} \leq q_i y_i & \qquad \forall i \in V_1 \\
\sum_{j \in V_2} x_{ij} \geq m_i y_i & \qquad \forall i \in V_1 \\
\sum_{i \in V_1} x_{ij} = d_j & \qquad \forall j \in V_2 \\
y_i \in \{0, 1\} & \qquad \forall i \in V_1 \\
x_{ij} \geq 0 & \qquad \forall i \in V_1, \forall j \in V_2
\end{aligned}
$$

Concave piecewise linear facility set-up costs

Assume that

$$F_i(u_i) = \begin{cases} 0 & \text{if } u_i = 0 \\ f_i' + g_i' u_i & \text{if } 0 < u_i \leq \overline{u}_i \\ f_i'' + g_i'' u_i & \text{if } u_i > \overline{u}_i \end{cases}$$

where $f_i' < f_i''$ and $g_i' > g_i''$.

We can model this situation by replacing each facility i with as many facilities as the number of intervals in the definition of $F_i(u_i)$, each one with fixed and variable facility costs and with lower and upper bounds on the activity level.

At optimality no more than one of the artificial facilities corresponding to the same actual facility is selected.

Single source constraints

Single-source constraints impose that each customer $j \in V_2$ be completely served by a single facility $i \in V_1$.

Example (Cortinhal and Captivo, 2003): SSCFLP - Single source capacitated facility location problem (single echelon, single commodity, linear transportation costs, fixed facility costs):

$$\begin{aligned}
\text{minimize } z = & \sum_{i \in V_1} \sum_{j \in V_2} c_{ij} x_{ij} + \sum_{i \in V_1} f_i y_i \\
\text{s.t. } & \sum_{j \in V_2} d_j x_{ij} \leq q_i y_i & \forall i \in V_1 \\
& \sum_{i \in V_1} x_{ij} = 1 & \forall j \in V_2 \\
& y_i \in \{0, 1\} & \forall i \in V_1 \\
& x_{ij} \in \{0, 1\} & \forall i \in V_1, \forall j \in V_2
\end{aligned}$$

The problem is formulated as a 0-1 linear programming model.

SSCFLP: Lagrangean relaxation

Constraints $\sum_{i \in V_1} x_{ij} = 1 \ \ \forall j \in V_2$ are dualized, i.e. they are transformed into penalty terms in the Lagrangean objective function:

$$z_{LR}(\lambda) = \sum_{i \in V_1} \sum_{j \in V_2} (c_{ij} - \lambda_j) x_{ij} + \sum_{i \in V_1} f_i y_i + \sum_{j \in V_2} \lambda_j,$$

where the coefficients $\lambda \geq 0$ are the Lagrangean multipliers.

The constraint set is now:

$$\begin{aligned}
\text{s.t. } & \sum_{j \in V_2} d_j x_{ij} \leq q_i y_i && \forall i \in V_1 \\
& \sum_{i \in V_1} q_i y_i \geq \sum_{j \in V_2} d_j && \\
& y_i \in \{0, 1\} && \forall i \in V_1 \\
& x_{ij} \in \{0, 1\} && \forall i \in V_1, \forall j \in V_2
\end{aligned}$$

For any choice of $\lambda \geq 0$, z^*_{LR} can be found by solving $|V_1| + 1$ instances of the binary knapsack problem and it is guaranteed to be a valid lower bound to z^*.

SSCFLP: dual optimization

The Lagrangean multipliers are iteratively adjusted by sub-gradient optimization.

$$\lambda_j^{(t+1)} = \lambda_j^{(t)} + \rho^{(t)} \frac{\overline{z} - z^*_{LR}(\lambda^{(t)})}{||g^{(t)}||} \quad \forall j \in V_2,$$

where

- $\overline{z}$ is the best incumbent upper bound;
- $z^*_{LR}(\lambda^{(t)})$ is the current optimal value of the Lagrangean objective;
- $g^{(t)}$ is the current subgradient: $g_j = 1 - \sum_{i \in V_1} x_{ij}$;
- $\rho^{(t)}$ is a step parameter, halved after every 5 iterations without improvement of the lower bound.

Stop when

- $\overline{z} - z^*_{LR}(\lambda^{(t)}) < 1$
- max. n. of iterations reached
- $||g^{(t)}|| = 0$

SSCFLP: Lagrangean heuristic

At each iteration of the subgradient algorithm, V_2 is partitioned into three subsets:

- $V_2^= = \{j \in V_2 : \sum_{i \in \mathcal{V}_1} x_{ij} = 1\}$
- $V_2^- = \{j \in V_2 : \sum_{i \in \mathcal{V}_1} x_{ij} = 0\}$
- $V_2^+ = \{j \in V_2 : \sum_{i \in \mathcal{V}_1} x_{ij} > 1\}$

Heuristic:

- Optimally reassign each $j \in V_2^- \cup V_2^+$ selecting a single source $i \in V_1$ for it;
- If the capacity constraint is violated, start a local search algorithm.

SSCFLP: exact optimization

Provably optimal solutions can be computed by branch-and-bound, where lower bounds are computed for each sub-problem with Lagrangean relaxation.

A suitable branching policy is needed. For instance (binary branching):

- branch on a y_i variable;
  - $y_i = 0$ in one branch;
  - $y_i = 1$ in the other.
- branch on a pair (u, v) of customers:
  - $x_{iu} = x_{iv}\ \forall i \in V_1$ in one branch;
  - $x_{iu} + x_{iv} \leq 1\ \forall i \in V_1$ in the other.