1 The facility location problem

A company is looking to determine locations for warehouses in order to supply its customers efficiently. It has several possible locations for the warehouses. There is a cost for operating a warehouse and there are shipping costs to supply the customers. This is an example of a facility location problem.

Facility location problems arise in many other situations. For example, we may consider locating firehouses to ensure a rapid response to any fire. Or we may want to locate processing facilities for corn, which should be close to the farms where the corn is grown.

Possible facility location

Customer

\(f_i\): cost of opening facility \(i\)

\(c_{ij}\): cost to ship one unit from facility \(i\) to customer \(j\)

\(d_j\): demand of customer \(j\)

A feasible solution:

Open facility

Customer

\(f_i\): cost of opening facility \(i\)

\(c_{ij}\): cost to ship one unit from facility \(i\) to customer \(j\)

\(d_j\): demand of customer \(j\)
2 Integer optimization formulation

Let $m$ be the number of facility locations under consideration. Let $n$ be the number of customers. We define two sets of variables:

$$y_i = \begin{cases} 
1 & \text{if location } i \text{ is opened} \\
0 & \text{otherwise} 
\end{cases}$$

$$x_{ij} = \text{amount of material shipped from location } i \text{ to customer } j$$

Objective function: min$_{x,y}$ $\sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$.

Constraints:

(a) Meet demand: $\sum_{i=1}^{m} x_{ij} = d_j$ for $j = 1, \ldots, n$.

(b) Can only ship from open facilities: $x_{ij} \leq d_j y_i$ for $i = 1, \ldots, m$, $j = 1, \ldots, n$.

Thus, if $y_i = 0$ then we must have $x_{ij} = 0$ for all $j = 1, \ldots, n$.

And if $y_i = 1$ then the constraint becomes $x_{ij} \leq d_j$, which is redundant.

Complete model:

$$\min_{x \in \mathbb{R}^{mn}, y \in \mathbb{R}^m} \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$
subject to

$$\sum_{i=1}^{m} x_{ij} = d_j \quad \text{for } j = 1, \ldots, n$$
$$x_{ij} \leq d_j y_i \quad \text{for } i = 1, \ldots, m, j = 1, \ldots, n$$
$$x_{ij} \geq 0 \quad \text{for } i = 1, \ldots, m, j = 1, \ldots, n$$
$$y_i \text{ binary} \quad \text{for } i = 1, \ldots, m$$

We did not impose any capacity limits on the facilities, so this problem is known as an uncapacitated facility location problem. If each location $i$ can only ship $u_i$ material, we get a capacitated facility location problem:

$$\min_{x \in \mathbb{R}^{mn}, y \in \mathbb{R}^m} \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$
subject to

$$\sum_{i=1}^{m} x_{ij} = d_j \quad \text{for } j = 1, \ldots, n$$
$$\sum_{j=1}^{n} x_{ij} \leq u_i \quad \text{for } i = 1, \ldots, m$$
$$x_{ij} \leq d_j y_i \quad \text{for } i = 1, \ldots, m, j = 1, \ldots, n$$
$$x_{ij} \geq 0 \quad \text{for } i = 1, \ldots, m, j = 1, \ldots, n$$
$$y_i \text{ binary} \quad \text{for } i = 1, \ldots, m$$

3 Aggregated model

Formulation (1) has $mn + n$ constraints. We can aggregate the constraints on shipping from open facilities to give a formulation with fewer constraints:

$$\sum_{j=1}^{n} x_{ij} \leq My_i \quad \text{for } i = 1, \ldots, m,$$
where $M$ is a large enough constant. When $y_i = 0$, this constraint forces each $x_{ij} = 0$. We must have $M$ large enough so that the constraint is redundant when $y_i = 1$. The maximum amount we could possibly ship from location $i$ is the sum of all the demands. Thus, it suffices to take $M = \sum_{j=1}^{n} d_j$. This gives a model:

$$\min_{x \in \mathbb{R}^{mn}, y \in \mathbb{R}^m} \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$
subject to
$$\sum_{i=1}^{m} x_{ij} = d_j \quad \text{for } j = 1, \ldots, n$$
$$\sum_{j=1}^{n} x_{ij} \leq \left(\sum_{j=1}^{n} d_j\right) y_i \quad \text{for } i = 1, \ldots, m$$
$$x_{ij} \geq 0 \quad \text{for } i = 1, \ldots, m, j = 1, \ldots, n$$
$$y_i \text{ binary} \quad \text{for } i = 1, \ldots, m$$

4 Comparing the models

We solve integer optimization problems by solving a sequence of linear relaxations, in either a branch-and-bound scheme or a cutting plane scheme or a combination. The aggregated formulation (2) has fewer constraints, which means that each relaxation is typically solved more quickly. However, this advantage is heavily outweighed by the observation that the relaxation of (2) is far weaker than the relaxation of (1). This means that far more linear optimization problems are solved if we use (2) than if we use (1).

Thus, the disaggregated formulation (1) is preferable to the aggregated formulation (2).

Consider the following example with two possible facility locations and three demand points:

The disaggregated formulation (1) has the constraints:

$$x_{11} \leq y_1, \ x_{12} \leq y_1, \ x_{13} \leq y_1, \ x_{21} \leq y_2, \ x_{22} \leq y_2, \ x_{23} \leq y_2,$$

while the aggregated formulation has the constraints

$$x_{11} + x_{12} + x_{13} \leq 3y_1, \ x_{21} + x_{22} + x_{23} \leq 3y_2.$$
The linear optimization relaxation of \((1)\) gives the optimal integral solution of value 16, with \(y_1 = 1\) and \(y_2 = 0\). By contrast, the linear optimization relaxation of \((2)\) gives a fractional solution with value \(13\frac{1}{3}\) with \(y_1 = \frac{1}{3}\) and \(y_2 = \frac{2}{3}\).

**The uncapacitated facility location problem and CPLEX**

This issue of a weak relaxation is now well-understood. Consequently, it is something that is identified by CPLEX and other solvers, and the formulation is strengthened automatically in the presolver. We can turn off the presolver by using the following command in AMPL:

```
AMPL: option cplex_options 'mipcuts=-1';
```

Running the two formulations with the presolver turned on and off on randomly generated problems with \(m = 20, 40, 60, 80, 100\) and \(n = 3m\) gives the following results. The “time” is in seconds on a 2008 Mac Pro, the “nodes” is the number of nodes in the branch-and-bound tree, and “simplex” is the total number of simplex iterations required.

<table>
<thead>
<tr>
<th>(m)</th>
<th>(m = 20)</th>
<th>(m = 40)</th>
<th>(m = 60)</th>
<th>(m = 80)</th>
<th>(m = 100)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time ()</td>
<td>nodes ()</td>
<td>simplex ()</td>
<td>time ()</td>
<td>nodes ()</td>
</tr>
<tr>
<td></td>
<td>(0.104137)</td>
<td>(0)</td>
<td>(480)</td>
<td>(0.269296)</td>
<td>(52)</td>
</tr>
<tr>
<td></td>
<td>(4.13291)</td>
<td>(114)</td>
<td>(14048)</td>
<td>(7.65573)</td>
<td>(865)</td>
</tr>
<tr>
<td></td>
<td>(24.1969)</td>
<td>(77)</td>
<td>(16998)</td>
<td>(257.869)</td>
<td>(25431)</td>
</tr>
<tr>
<td></td>
<td>(393.583)</td>
<td>(1461)</td>
<td>(444491)</td>
<td>out of memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(4297.36)</td>
<td>(11001)</td>
<td>(3505958)</td>
<td>out of memory</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(m)</th>
<th>(m = 20)</th>
<th>(m = 40)</th>
<th>(m = 60)</th>
<th>(m = 80)</th>
<th>(m = 100)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time ()</td>
<td>nodes ()</td>
<td>simplex ()</td>
<td>time ()</td>
<td>nodes ()</td>
</tr>
<tr>
<td></td>
<td>(0.07881)</td>
<td>(0)</td>
<td>(441)</td>
<td>(1.23681)</td>
<td>(4871)</td>
</tr>
<tr>
<td></td>
<td>(4.82521)</td>
<td>(144)</td>
<td>(16408)</td>
<td>(1919.65)</td>
<td>(3432040)</td>
</tr>
<tr>
<td></td>
<td>(27.297)</td>
<td>(348)</td>
<td>(58013)</td>
<td>out of memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(371.534)</td>
<td>(1421)</td>
<td>(433540)</td>
<td>out of memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(3669.83)</td>
<td>(8348)</td>
<td>(2912548)</td>
<td>out of memory</td>
<td></td>
</tr>
</tbody>
</table>

Formulation \((1)\) is clearly better. Formulation \((2)\) is improved by the presolver, but it’s still worse than \((1)\).