

Input: $n \times n$ matrix D giving distances between n points $C = \{c_1, \dots, c_n\}$. The distances form a metric, i.e. $d_{ii} = 0$, $d_{ij} = d_{ji}$, $d_{ij} \leq d_{ik} + d_{kj}$ for all i, j, k . Costs f_1, \dots, f_n of constructing a facility at c_1, \dots, c_n .

The points C are locations of clients which need the facilities. A facility can be opened (constructed) at location c_i at cost f_i . Each client must be connected to an opened facility at a connection cost equal to the distance between the client and the facility. We may construct as many facilities as we want, so that the total cost of constructing all facilities and connecting each client to some facility is minimized. A facility can serve an unbounded number of clients; variations in which there is a bound have also been studied. Because there is no bound, each client is best connected to the closest open facility.

Output: Subset $S \subseteq C$ such that $\sum_{c_j \in S} f_j + \sum_i \min_{c_j \in S} d_{ij}$ is minimized.

If the distances are not required to form a metric, then it can easily be seen that this problem is a generalization of the set cover problem. Since set cover is known to be hard to approximate beyond a log factor, the same holds of this problem. Also, a greedy strategy only slightly more complicated than set cover can be used to get a log factor approximation. In other words, the non-metric case is well understood.

For the metric case, which is our subject, we will show a 6 approximation. Much better, but more complex, approximations are known in the literature.

1 IP Formulation and LP relaxation

The integer programming formulation uses two kinds of indicator variables. First we have variables $y_i, i = 1, \dots, n$ which indicate whether or not a facility is to be opened at location c_i . We also have indicator variables x_{ij} denoting whether or not client at c_i is to be served by a facility at c_j . Of course, we must ensure that we set $x_{ij} = 1$ only if a facility is opened at c_j , i.e. only if $y_j = 1$. The objective function is:

$$\text{minimize} \quad \sum_j f_j y_j + \sum_{ij} d_{ij} x_{ij}$$

We ensure that every client location c_i is connected to some facility:

$$\forall i \quad \sum_j x_{ij} = 1$$

And a client is connected to a facility only if it is opened:

$$\forall i, j \quad x_{ij} \leq y_j$$

Notice that asserting $p \leq q$ for indicator variables is equivalent to asserting $p \Rightarrow q$, which is what we wanted. We require all our variables to be either 0 or 1:

$$\forall i, j \quad x_{ij} \in \{0, 1\}, \quad y_j \in \{0, 1\}$$

To get an LP relaxation, instead of the above condition, we have the following:

$$\forall i, j \quad 0 \leq x_{ij} \leq 1, \quad 0 \leq y_j \leq 1$$

2 Algorithm

The first step in the algorithm is to solve the LP relaxation. This can be done in polynomial time in n , given that we have $O(n^2)$ variables and $O(n^2)$ constraints.

Suppose x', y' denote the solutions to the LP relaxation, and x^*, y^* the solutions to the IP. Then we know that the objective value for the LP is at most that for the IP:

$$\sum_j f_j y'_j + \sum_{ij} d_{ij} x'_{ij} \leq \sum_j f_j y^*_j + \sum_{ij} d_{ij} x^*_{ij}$$

Thus we have a lower bound on the cost of the optimal solution, which is the first benefit of solving the LP relaxation.

Before rounding the LP, it is useful to see if the LP solution can be interpreted. Clearly, the first term $\sum_j f_j y'_j$ can still be thought of as the connection cost, but if $y'_j = 0.5$ we think of facility j as “half-opened”. The second term, $\sum_{ij} d_{ij} x'_{ij}$ can be interpreted as the connection cost, with $d'_i = \sum_j d_{ij} x'_{ij}$ the connection cost for client i . Also, $x'_{ij} = 0.1$ could be interpreted as client i gets a tenth of its service from facility j (which is guaranteed to be opened at least to extent 0.1 by the LP condition $x'_{ij} \leq y'_j$). Since expression $d'_i = \sum_j d_{ij} x'_{ij}$ and $\sum_j x'_{ij} = 1$, we can think of d'_i as a weighted average of the various connection costs client i pays, or the distances to which it is connected.

The next idea is to identify the set of “near” facilities N_i for a client i . These simply are the facilities that are at a distance at most $2d'_i$. The good news about such facilities is that they give enough service to i .

Lemma 1 *For any client i , $\sum_{j \in N_i} x'_{ij} \geq 1/2$.*

Proof: Just some basic ideas about averaging. $d'_i = \sum_j d_{ij} x'_{ij} \geq \sum_{j \notin N_i} (2d'_i) x'_{ij} = 2d'_i \sum_{j \notin N_i} x'_{ij}$. Thus $\sum_{j \notin N_i} x'_{ij} \leq 1/2$. Thus the result follows by taking the complement. ■

So we can now construct a new solution x, y to the LP. This solution will not be integral (and hence is often called fractional) but will have the desirable feature that each client is only connected to near facilities.

1. For each i, j we set $x_{ij} = 0$ if $j \notin N_i$. For $j \in N_i$ we tentatively set $x_{ij} = 2x'_{ij}$.
2. For all i , if $\sum_i x_{ij} > 1$, we arbitrarily reduce some of the x_{ij} so that the sum becomes 1.
3. For all j , we then set $y_j = \max_i x_{ij}$, so that inequalities $x_{ij} \leq y_j$ are also satisfied.

Lemma 2 x, y constitute a feasible solution to the LP. Further its cost is at most twice that of the solution x', y' , i.e. $\sum_j f_j y_j \leq 2 \sum_j f_j y'_j$ and $\sum_{ij} d_{ij} x_{ij} \leq 2 \sum_{ij} d_{ij} x'_{ij}$. Also, $x_{ij} > 0 \Rightarrow d_{ij} \leq 2d'_i$.

Proof: We know that $\sum_{j \in N_i} x'_{ij} \geq 1/2$. Thus after step 1 of our construction, we have $\sum_j x_{ij} \geq 1$ for all i . In step 2 we reduce some x_{ij} in case the sum exceeded 1, so at the end of step 2, we have $\sum_j x_{ij} = 1$ for all i as required. At the end of step 3 we have $x_{ij} \leq y_j$ for all i, j , thus ensuring that x, y is feasible.

Client i is connected at most a distance $2d'_i$ away by construction. Hence the connection cost is at most $2d'_i$. We know that for some i we have $y_j = x_{ij}$. But $x_{ij} \leq 2x'_{ij} \leq 2y'_j$. Thus $y_j \leq 2y'_j$. Thus the facility opening cost has also at most doubled. ■

We have paid a price of a factor of 2, but we have some flexibility in rounding: if $x_{ij} > 0$ we can round it up to 1 without worrying whether we are paying too much in connection cost. So now we stop worrying about connection costs.

The rounding step is:

1. Declare all clients to be unsatisfied.
2. Let i denote the unsatisfied client that has smallest d'_i .
3. Let k denote the least expensive open facility in N_i .
4. Open k fully, i.e. set $y_k = 1$, and for all $j \neq i$, set $y_j = 0$.
5. Consider all unsatisfied clients q connected to any $j \in N_i$, i.e. such that $x_{qj} > 0$. Set $x_{qk} = 1$, and $x_{qp} = 0$ for $p \neq k$. Declare each such q satisfied.
6. Declare i to be satisfied.
7. If any unsatisfied clients remain, then repeat from step 2.

Let us analyze any single iteration. We know that the chosen unsatisfied client i is fully served at the time it is chosen. Thus $1 = \sum_j x_{ij} \leq \sum_{j \in N_i} y_j$. The cost of opening the facilities in N_i is $\sum_{j \in N_i} f_j y_j \geq \sum_{j \in N_i} f_k y_j \geq f_k$ because

$\sum_{j \in N_i} y_j \geq 1$. Thus by opening k and fully closing down the facilities in N_i we could only be reducing the opening cost.

Consider a client c_q that gets connected to the facility at c_k . $d_{qk} \leq d_{qj} + d_{ji} + d_{ik}$. But $d_{qj} \leq 2d'_q$. Also $d_{ji}, d_{ik} \leq 2d'_i \leq 2d'_q$, the second inequality arises because we chose i so that d'_i is smallest. Thus $d_{qk} \leq 6d'_q$. Also $d_{ik} \leq 2d'_i$.

Thus in each iteration, we are not increasing the construction cost. So the final construction cost is the same as the construction cost before the rounding step, i.e. twice the cost in the LP. The connection cost for each client i is at most $6d'_i$, i.e. grown to at most 6 fold. Thus overall this is a 6 approximation algorithm.

Exercises

1. We defined “near” facilities to be those within distance $2d'_j$. Define this more carefully to get a factor 4 algorithm.
2. Suppose the facilities and clients all lie on a straight line. Devise an optimal polytime algorithm for this problem. Hint: dynamic programming.