CS 684: Approximation and Network Algorithms	**11/09/01**
Access Network Design	**scribe: Vladimir Dizhoor**

1 Gathering problem review

The Gathering Problem is a version of the Facility Location:
We are given:

- A set of potential facilities
- A set of clients with connection cost $c_{ij} \geq 0$
- $\alpha > 0$

We want to open some facility f, assigning each client j to an open facility i_j such that

- each open facility must have at least α clients
- the total assignment cost $\sum_{j \in Clients} c_{j,i_j}$ is minimized

Last time we gave a solution to this problem with a total cost $\leq$ 6 times the OPT gathering cost, such that at the same time each open facility has $\geq \frac{\alpha}{3}$ clients assigned to it.

In general, if we use an r-approximation algorithm for th efacility location problem, we $q > 1$ is some constant, we can extend this algorithm so that the resulting cost is $\leq r\frac{q+1}{q-1}$ of the OPT assignment cost and each open facility has $\geq \frac{\alpha}{q}$ clients assigned to it.

2 Access network design

We are given:

- a graph $G = (V, E)$ with $c_e \geq 0$ for every $e \in E$
- an *access* point $a \in V$
- clients $S \subseteq V$
- $\alpha > 0$

What we want is to find paths P_s from each $s \in S$ to the access point a such that the quantity

$$\sum_{e \in E} c_e \cdot min(\alpha, \text{number of paths using } e)$$

is minimized.

The optimum structure has a tree T containing node a. Every edge of T is used by $> \alpha$ paths. We pay $\sum_{e \in T} \alpha c_e$ and we also cover the min-cost paths from each $s \in S$ to T.

3 Approximation algorithm for access network design

For an approximate solution to the problem we solve an instance of the Gathering Problem. We set the set of facilities to be all the nodes in the graph. We keep α as given. Connection costs are the min cost paths. The clients are a slightly modified set S. In order to guarantee that node a is chosen by the gathering algorithm, we create α copies of node a. Thus, $Clients = S + \alpha$ copies of node a.

We find a min-cost Steiner Tree on open facilities.

The result feasible of cost $\leq$ cost of gathering $+ \sum_{e \in T} \alpha c_e$. Here T is the Steiner Tree found. We need to compare the Steiner Tree and the gathering to the OPT of network design. Recall from last lecture the structure of an optimal solution. The edges that have more than α paths through them form a spanning tree, and all clients connect to this tree via a shortest path. We can view the shortest paths part of this solution as the "gathering part" of the designed network.

Claim 3.1 *Gathering Solution OPT $\leq$ Network Design OPT. In fact, even $\leq$ Gathering part of the network design OPT.*

What is the OPT cost of the Steiner Tree problem that we solve? To simplify thinking, at first assume (incorrectly) that we gather $\geq \alpha$ at all facilities except maybe a.

Claim 3.2 *Min-cost Steiner Tree $\leq$ Gathering cost + OPT of original problem*

Proof: Consider an access network instance where all clients move to their gathering point (facilities). In this case the optimum network design is the OPT Steiner tree. In other words, the Min-cost Steiner tree is the OPT of access network design where all clients moved to their facility and it is $\leq$ than the cost of solution path for each client j assigned to client i_j. Use path $C_j - j$ followed by j's optimum path. The cost of this solution is $\leq$ gathering cost + the opt of original problem.

Now, in reality at least $\frac{\alpha}{3}$ are gathered as opposed to α which was assumed previously.

Claim 3.3 *Min-Cost Steiner tree $\leq$ 3 Gathering cost + 3 OPT of original problem*

Proof: Min-cost Steiner tree $\leq$ 3 OPT of the modified network design. The rest goes automatically as above accept for the previously unused factor of 3.

Theorem 3.4 *Algorithm proposed is an O(1) approximation algorithm. The constant is 48.*
Proof: algorithm's gathering cost is $\leq$ 6 OPT gathering cost $\leq$ 6 OPT
algorithm's Steiner tree cost $\leq$ 2 OPT Steiner tree + 6 OPT
Thus the algorithm's cost is bounded by 48 of OPT.

4 Network design (general)

We are given

- graph $G = (V, E)$
- pairs (s_i, t_i)
- a $c_e \geq 0$ associated with edges
- α as before

We need to find paths p_i from s_i to t_i. It is cheaper if edges are being used simultaneously from many paths. Thus, as before the cost of the solution is

$$\sum_{e \in E} c_e \cdot min(\alpha, \text{number of paths using } e)$$

and we seek to minimize it.

The best known approach provides an $O(log(n) \cdot log(log(n))$ approximation ratio. Here $n = |V|$. Note a special case of the problem occurs when G is a tree, since there is only one path from s_i to t_i in a tree (there is only one path between any two nodes in a tree).

Another (quite uncommon) case occurs when the shortest path metric is a tree metric. That is, *every* shortest path goes through *one* tree. In this case, we would also use shortest paths, sharing edges as much as possible. When ties between shortest paths occur, we exchange any path for one using the metric defining tree contained within the graph.

5 Bartal Theorem

There is a useful result that bounds a tree metric relative to any other metric defined on a set of points. Let us denote $dist_T(i, j)$ to be the distance between points i and j using the tree (T) metric, and let $dist(i, j)$ mean a distance defined in any other metric. If we had a tree T whose distances are very close to the actual distances we coul duse the tree in our approximation algorithm. Say, if we had that

$$dist(i, j) \leq dist_T(i, j) \leq O(log(n) \cdot log(log(n))) \cdot dist(i, j)$$

then we solve the network design on a tree optimally and obtain a $O(log(n) \cdot log(log(n)))$ approximation.

However, we cannot expect to have such a tree for any metric. However, Bartal proved that if we chose the tree at random we can do this well in expectation. To see an example of how this works, consider a simple long cycle of n nodes. Now select an edge at random and make $T =$ cycle without that edge. This tree is quite bad in the above sense as the distance between the ends of the deleted edge is 1 in the cycle, but $n - 1$ in the tree. However, if we choose the edge at random, then the expected distance

$$dist_T(i, j) = \text{probability that edge } (i, j) \text{ is not selected} \cdot 1 + \text{probability that edge } (i, j) \text{ is selected} \cdot (n - 1)$$

$$dist_T(i, j) = \frac{n-1}{n} \cdot 1 + \frac{1}{n} \cdot (n - 1) \leq 2$$

Bartal proved that in general, there exist such trees $T_1, T_2, \ldots, T_k$ with corresponding probabilities $p_1, p_2, \ldots, p_k$, s.t. $\sum p_i = 1$, so

$$dist(i,j) \leq \mathrm{Exp}_T(dist_T(i,j)) \leq O(log(n) \cdot log(log(n))) \cdot dist(i,j)$$

Thus the approach to network design consists of the following:

- finding min-cost paths metric on V.
- using Bartal's theorem.
- selecting a tree ar random T_k with probability p_k and routing on tree T_k.