CMSC 858K Advanced Algorithms	Lecture 03
Lecturer: Samir Khuller	Feb 4, 1999

Notes by Samir Khuller.

11 Approximation Algorithms: K-Centers

The basic K-center problem is a fundamental facility location problem and is defined as follows: given an edge-weighted graph $G = (V, E)$ find a subset $S \subseteq V$ of size at most K such that each vertex in V is "close" to some vertex in S. More formally, the objective function is defined as follows:

$$\min_{S \subseteq V} \max_{u \in V} \min_{v \in S} d(u, v)$$

where d is the distance function. For example, one may wish to install K fire stations and minimize the maximum distance (response time) from a location to its closest fire station. The problem is known to be NP-hard. See Section 9.4 for more details.

11.1 Gonzalez's Algorithm

Gonzalez describes a very simple greedy algorithm for the basic K-center problem and proves that it gives an approximation factor of 2. The algorithm works as follows. Initially pick any node v_0 as a center and add it to the set C. Then for $i = 1$ to K do the following: in iteration i, for every node $v \in V$, compute its distance $d^i(v, C) = \min_{c \in C} d(v, c)$ to the set C. Let v_i be a node that is farthest away from C, i.e., a node for which $d^i(v_i, C) = \max_{v \in V} d(v, C)$. Add v_i to C. Return the nodes $v_0, v_1, \ldots, v_{K-1}$ as the solution.

We claim that this greedy algorithm obtains a factor of 2 for the K-center problem. First note that the radius of our solution is $d^K(v_K, C)$, since by definition v_K is the node that is farthest away from our set of centers. Now consider the set of nodes $v_0, v_1, \ldots, v_K$. Since this set has cardinality $K+1$, at least two of these nodes, say v_i and v_j, must be covered by the same center c in the optimal solution. Assume without loss of generality that $i < j$. Let R^* denote the radius of the optimal solution. Observe that the distance from each node to the set C does not increase as the algorithm progresses. Therefore $d^K(v_K, C) \leq d^j(v_K, C)$. Also we must have $d^j(v_K, C) \leq d^j(v_j, C)$ otherwise we would not have selected node v_j in iteration j. Therefore

$$d(c, v_i) + d(c, v_j) \geq d(v_i, v_j) \geq d^j(v_j, C) \geq d^K(v_K, C)$$

by the triangle inequality and the fact that v_i is in the set C at iteration j. But since $d(c, v_i)$ and $d(c, v_j)$ are both at most R^*, we have the radius of our solution $= d^K(v_K, C) \leq 2R^*$.

11.2 Hochbaum-Shmoys method

We give a high-level description of the algorithm. We may assume for simplicity that G is a complete graph, where the edge weights satisfy the triangle inequality. (We can always replace any edge by the shortest path between the corresponding pair of vertices.)

High-Level Description

The algorithm uses a thresholding method. Sort all edge weights in non-decreasing order. Let the (sorted) list of edges be $e_1, e_2, \ldots e_m$. For each i, let the threshold graph G_i be the unweighted subgraph obtained from G by including edges of weight at most $w(e_i)$. Run the algorithm below for each i from 1 to m, until a solution is obtained. (Hochbaum and Shmoys suggest using binary search to speed up the computation. If running time is not a factor, however, it does appear that to get the best solution (in practice) we should run the algorithm for all i, and take the best solution.) In each iteration, we work with the unweighted subgraph G_i. Since G_i is an unweighted graph, when we refer to the distance between two nodes, we refer to the number of edges on a shortest path between them. In iteration i, we find a solution using some number of centers. If the number of centers exceeds K, we prove that there is no solution with cost at most $w(e_i)$. If the number of centers is at most K, we find a solution.

Observe that if we select as centers a set of nodes that is sufficiently well-separated in G_i then no two centers that we select can share the same center in the optimal solution, if the optimal solution has radius $w(e_i)$. This suggests the following approach. First find a maximal independent set I in G_i^2. (G_i^2 is the graph obtained by adding edges to G_i between nodes that have a common neighbor.) Let I be the chosen set of centers. If I has cardinality at most K we are done, since each vertex has a neighbour in the independent set at distance $2w(e_i)$. This means that all vertices have a center close to them. If the cardinality of I exceeds K, then there is no solution with cost $w(e_i)$. This is because no two vertices in I can be covered by a common center. (If there was a vertex that could cover both the nodes in I at distance $w(e_i)$ then, in G_i, these two nodes have a common neighbor and cannot both be in a maximal independent set.

Since the optimal solution has some cost $\delta = w(e_j)$ for some j. The claim is that we will find a maximal independent set of size at most K when we try $i = j$ and build the graph G_i. This graph has a dominating set of size at most K. If $N(k)$ is the set of vertices dominated (either k or a vertex adjacent to k in G_j) by $k \in OPT$ (OPT is the optimal solution), then we can pick at most one vertex in any maximal independent set in $N(k)$. This implies that we will find a maximal independent set of size at most K in G_i^2. We might find one for some $i < j$, in which case we obtain a solution with cost $2w(e_i) \leq 2w(e_j)$.