CSC 2421 - Approximation Algorithms

Lecture #5: 5 February 2003

Lecturer: A. Borodin

Scribe Notes by: D. Cashman, P.A. Papakonstantinou

In the previous lecture we introduced the *Uncapacitated Facility Location* (U.F.L.) problem and we presented an algorithm based on the primal-dual paradigm, for the metric version of this problem. More specifically, we discussed a way to view the U.F.L. problem as a Set Cover problem, and modified the set cover algorithm so as to present a combinatorial algorithm for the metric UFL. The analysis of this algorithm is quite technical, so we will merely sketch the LP dual fitting proof for the approximation ratio of this algorithm. The second topic of this lecture is the *Steiner Tree* and the *Steiner Forest* problems. For the Steiner Forest problem we present an extension to the methodology for the primal-dual algorithms we have seen so far. The extension is that we add a "clean-up" step to the primal-dual algorithm.

1 Definitions Review

Definition Let $\mathcal{F}$ be a set of facilities and $\mathcal{C}$ a set of cities. Each facility $i \in \mathcal{F}$ has an *opening cost* f_i and there is also a *connection cost* c_{ij} for the connection of the j-th city to the i-th facility. The **Uncapacitated Facility Location** problem is to find a function $\phi : \mathcal{C} \to \mathcal{F}$ such that the total cost of connecting a city to an open facility plus the cost of the opened facilities is minimum.

The above problem has two general versions:

1. *Disjoint*: $\mathcal{F} \cap \mathcal{C} = \emptyset$
2. *Complete*: $\mathcal{F} = \mathcal{C}$

We focus on the disjoint version of the problem, but it can be easily shown that any α-approximation for the disjoint model implies an α-approximation for the complete model.

The algorithm presented in the previous lecture is for the *Metric* U.F.L. where the costs c_{ij} enjoy the triangle inequality.

Below we define a polynomial size integer program for the U.F.L. This formulation was used in the first constant approximation algorithms for the metric UFL.

minimize

$$\sum_{i \in \mathcal{F}} x_i f_i + \sum_{i \in \mathcal{F}, j \in \mathcal{C}} c_{ij} u_{ij}$$

subject to

$$u_{ij} \geq x_j,\ i \in \mathcal{F}, j \in \mathcal{C} \tag{1}$$
$$\sum_{i \in \mathcal{F}} u_{ij} \geq 1,\ j \in \mathcal{C} \tag{2}$$
$$x_i \in \{0,1\},\ \ u_{ij} \in \{0,1\} \tag{3}$$

where x_i, $i \in \{1,\ldots,n\}$ are indicator variables denoting that facility i is open or not, and $u_{i,j}$, $i,j \in \{1,\ldots,n\}$ are indicator variables denoting that the city j is connected to facility i. Equation (1) ensures that a city is connected to an open facility and equation (2) ensures that a city is connected to at least one facility. We can relax the above integer program to an LP, by changing equation (3) to $x_i \geq 0,\ \ u_{ij} \geq 0$. The dual of the above (primal) is:

maximize

$$\sum_{i \in \mathcal{C}} a_i$$

subject to

$$\begin{aligned} a_j - b_{ij} &\leq c_{ij},\ i \in \mathcal{F}, j \in \mathcal{C} \\ \sum_{j \in \mathcal{C}} b_{ij} &\leq f_i,\ i \in \mathcal{F} \\ a_j, b_{ij} &\geq 0 \end{aligned} \tag{4}$$

2 U.F.L. as a variation of the Set Cover Problem

We can easily reduce the Set Cover (optimization) problem to the non-metric version of the U.F.L. The *approximation preserving reduction* is by simply considering $\mathcal{F} = \mathcal{S}$ and $\mathcal{C} = \mathcal{U}$, where $\mathcal{S}$ is the set of sets, and $\mathcal{U}$ is the universe of the elements. Consider $f_i = cost(S_i)$, $S_i \in \mathcal{S}$ and $c_{ij} = 0$, if $e_j \in S_j$ and $c_{ij} = \infty$ otherwise. Approximating the (non-metric) U.F.L. is as hard as approximating the Set Cover, and thus (assuming $P \neq NP$) the approximation ratio is $\Omega(\log n)$, where n is the number of cities.

In what follows we are only concerned about the Metric U.F.L. The best known polytime approximation guarantee for this problem is 1.52, and when we consider only greedy algorithms the best known approximation ratio is 1.61. We present a greedy (i.e. an adaptive greedy priority) algorithm that solves the problem using a Set Cover approach and has approximation guarantee 1.861.

Definition We define as *star*, s, the pair $s = (i, C')$, where $C' \subseteq \mathcal{C}$, and $i \in \mathcal{F}$. Also, the a cost function from the set of stars to $\mathcal{Q}^{\geq 0}$ is: $c(s) = f_i + \sum_{j \in C'} c_{ij}$.

In an analogous way to the set cover algorithm (presented in the first lecture), we define a notion of cost-effectiveness:

Definition Given a star $s = (i, C')$ and a set of (covered) cities $C \subseteq \mathcal{C}$, let the r "uncovered" cities of C' have costs $c_{ij_1} \leq c_{ij_2} \leq \ldots \leq c_{ij_r}$. The cost-effectiveness of s, given C, is:

$$\frac{g_i + \sum_{j=1}^{r} c_{ij_s}}{r} \tag{5}$$

where $g_i = \begin{cases} f_i & \text{if facility } i \text{ is not opened} \\ 0 & \text{otherwise} \end{cases}$

The algorithm proceeds exactly like the Set Cover algorithm (based on cost-effectiveness), with the only modification, that as a star corresponding to facility i is opened, we set $f_i = 0$.

Metric U.F.L. Algorithm

1. Let $\mathcal{U}$ be the set of unconnected cities. In the beginning all cities are unconnected, $\mathcal{U} = \mathcal{C}$.
2. **while** $\mathcal{U} \neq \emptyset$
   (a) Among all stars find the most cost-effective (see definition) star (i, C'), open facility i if it is not already opened, and connect all cities in c' to the facility i.
   (b) Set $f_i \leftarrow 0, \mathcal{U} \leftarrow \mathcal{U} \backslash C'$

Remark Note that a facility i can be chosen again, but only the first time will f_i be non-zero in the cost-effectiveness calculation. Furthermore, notice that the "actual" number of stars is exponential to the number of cities. The reason is that each facility is associated to $\sum_{k=1}^{n} \binom{n}{k} = 2^n - 1$ sets of cities, where n is the number of cities. This is not a problem, because in the implementation of the algorithm, we do not explicitly construct the stars. For the i-th facility, we order the r uncovered cities in an increasing order $c_{ij_1} \leq c_{ij_2} \leq \ldots \leq c_{ij_r}$. It can easily be seen that the most cost-effective star will consist of a facility and a set, containing the first k cities, for some constant k less than or equal to the number of cities.

This algorithm has approximation guarantee 1.861. The analysis is quite technical and we do not present the proof.

Claim 1 *The above algorithm is a* 1.861 *approximation algorithm for the Metric U.F.L.*

The analysis uses dual-fitting , as was done for the Set Cover problem. The previous algorithm can be restated using the LP-formulation, in complete analogy to the greedy algorithm for the Set Cover problem. We will not restate the algorithm here (as it has a straightforward analogy to the one presented in lecture 3 for the Set Cover). Briefly: we raise "uniformly" all variables a_j. The contribution of each city goes towards opening at most one facility and connecting the city to an open facility. The total cost of the solution produced by the algorithm will be equal to the $\sum_{j \in \mathcal{C}} a_j$ of the "contributions". However, $(\vec{a}, \vec{b})$, where $b_{ij} = \max(a_j - c_{ij}, 0)$, is no longer a dual feasible solution. The reason is that $\sum_{j \in \mathcal{C}} \max(a_j - c_{ij}, 0)$ can be greater than f_i and hence the second constraint of the dual program is violated. However, if we show that for some $\gamma > 1$, $(\vec{a}/\gamma, \vec{b}/\gamma)$ is a feasible dual solution, then by the weak duality theorem, $\left(\sum_{j \in \mathcal{C}} a_j\right)/\gamma$ is a lower bound for the optimum solution, and therefore the approximation ratio of the algorithm is γ. For the analysis we choose $\gamma = 1.861$.

3 Steiner Tree and Steiner Forest problems

Definition Given a graph $G = (V, E)$ with a cost function $c : E \rightarrow \mathcal{Q}^{\geq 0}$, and a subset $V' \subseteq V$, the **Steiner tree problem** is to find a minimum cost subgraph $T \subseteq G$ such that $\forall u, v \in V'$, there exists a path in T from u to v. Nodes used in T that are not in V' are called *Steiner Nodes.*

Note that in the Steiner tree problem, T will always be a tree, since if there is a cycle, at least one edge can be removed while retaining all paths between vertices in V', and if there are two unconnected components, one of them must not contain any vertices in V' (because otherwise there would not be a path between some pair of vertices in V'), and can therefore be deleted.

The Steiner forest problem extends the problem to include multiple sets of vertices that must be connected. That is, given the above graph definition, and $V_1, V_2, \ldots, V_k, V_i \subseteq V$, the **Steiner forest problem** is to find a minimum cost subgraph $T \subseteq G$ such that $\forall i, \forall u, v \in V_i$, there exists a path in T from u to v. In this case T may be a set of trees, i.e. a forest.

3.1 Approximation of the Steiner Tree problem

One may assume that c is a metric[1] for the purpose of evaluating the approximation factor of any algorithm to solve the Steiner tree problem.

Theorem 2 *An approximation algorithm for the Metric Steiner Tree is also an approximation algorithm for the non-metric Steiner Tree problem, with the same approximation ratio.*

Proof Given a graph $G = (V, E)$ and a cost function c that is not a metric, we construct a graph $G' = (V, E')$ and metric cost function c' as follows:

For each pair of nodes $u, v \in V$, the cost of the edge in G' connecting u and v is the cost of the shortest path in G between u and v. If there is no path in G, then there is no edge between u and v in G'. Note that in the Steiner tree problem, this will necessarily result in a complete graph.

Given a solution to the Steiner tree problem in G', for each edge e' in the solution, find the corresponding path in G, and include each of those edges in the solution, if it has not already been included. An edge is added only if it does not create a cycle. Since the cost of e' is the cost of that path, the solution in G will have cost less than or equal to the the solution in G'.

However, the optimal solution OPT in G must always have total cost at least that of the solution OPT' in G'. This is true because any edge in G has a corresponding edge in G' with the same or smaller cost (because at worst, that edge in G may be taken as the path for finding the cost in G'). Thus, any tree in G may be used in G', but it will have the same or lower cost, and therefore the optimal solution in G corresponds also to G' with the same or lower cost.

Thus, the cost of the optimal solution in G is always the same or larger than the cost of the optimal solution in G', but given any solution in G', a solution with the same or smaller cost can be found in G. So, the approximation factor of the given solution in G' must be the same or greater than the approximation factor for the original graph G. Clearly, the above transformation can be works within polynomial time. Thus, the approximation factor for a polynomial algorithm that solves the metric Steiner tree problem must also hold for the general Steiner tree problem, completing the proof.

As it happens, the same result holds for the Steiner forest problem, and the argument is similar, but since we will not be making use of that fact, the proof is left as an exercise.

Given the result just proved, we now assume a metric cost function c, and consider the Steiner tree problem. Specifically, we will find a 2-approximation to the problem by finding the minimum spanning tree (MST) of V' (which can be done in polynomial time).

Theorem 3 *The cost of an MST on V' is a 2-approximation of the optimal Steiner tree.*

[1] Recall that the definition of a metric function (for the case of the graph vertices) is that for any vertices u, v, w, (1) $c(u, v) \leq c(u, w) + c(v, w)$, (2) $c(u, v) = c(v, u)$, (3) $c(u, v) \geq 0$.

Proof Consider a Steiner tree of cost OPT. Double all of the edges of the tree (i.e. we have two copies of each edge $\{u, v\}$), and obtain an Eulerian cycle connecting all of the vertices of V', as well as the Steiner nodes. This is possible because the degree of every vertex is now even. The cost of this path is at most $2 \cdot OPT$ (i.e. the cost of all of the edges in the doubled tree). Now construct a Hamiltonian cycle on the vertices of V' by tracing the cycle, but for each vertex that is a Steiner node, or that has already been visited, skip over it, and connect the previous node directly to the next non-Steiner, unvisited node in the cycle (create shortcuts). This is guaranteed not to increase the cost beyond $2 \cdot OPT$, because of the assumption that c is a metric. Deleting one edge of this Hamiltonian cycle gives a spanning tree on V', whose cost is at most $2 \cdot OPT$. Hence, the MST (which has cost less than or equal to any other spanning tree) must also have cost at most $2 \cdot OPT$.

The 2-approximation bound for this algorithm can be shown to be (almost) tight by considering a graph with $n + 1$ nodes. n of the nodes are in the set V', and all of them have cost 2 between each other, and cost 1 between them and the last node (which we will call i). The optimal solution connects each node to node i, for a cost of n. The MST on V', though, cannot include any of the cost 1 edges (since they all connect to node i, which is not in V'), and must have $n - 1$ edges (since it is a tree with n nodes). Hence, the cost of the MST is $2(n - 1)$, and the approximation ratio is $2(n - 1)/n$, which approaches 2 for large n.

Better approximation ratios are known for the Steiner tree problem. I think the best known poly time approximation ratio is 1.55 for the Steiner tree porblem whereas the 2-spproximation for Steiner forest that will now follow is the best known.

3.2 Approximating the Steiner Forest problem

We will now turn to finding a good approximation algorithm for the Steiner forest problem. The Steiner forest problem can be viewed as a special case of the hitting set problem. Recall the definition of the hitting set problem:

Definition Given subsets $T_1 \dots T_q$, ground set E, and cost $c : E \to \mathcal{Q}^{\geq 0}$, find a minimum cost $S \subset E$, such that for all i, $A \bigcap T_i \neq \emptyset$.

For each possible cut $(S, \overline{S})$ of the vertices in G, define the following:

$$\delta(S) = \{e | e = (x, y), x \in S, y \in \overline{S}\}$$

$$f(S) = \begin{cases} 1 & \text{if } \exists i, u \in S, v \in \overline{S},\ \ u, v \in V_i \\ 0 & \text{otherwise} \end{cases}$$

Note that if $f(S) = 1$, then there must be an edge across the cut for a valid Steiner forest, and if $f(S) = 0$, there does not need to be one. The LP formulation for this problem is then as follows:

minimize

$$\sum_{e \in E} c_e x_e$$

subject to

$$\sum_{e \in \delta(S)} x_e \geq f(S), S \subseteq V$$

In the IP program, $x_e \in \{0,1\}$, and we are simply requiring that there is some edge between every cut that needs to be bridged. In the LP program, we relax this constraint and allow $x_e \geq 0$. Note that the number of possible cuts, and therefore the size of the LP problem, is exponential in the number of vertices. However, the algorithm used will address the constraints in a way that still allows a polynomial run-time.

The dual program is:

maximize

$$\sum_{S \subseteq V} f(S) \cdot y_S$$

subject to

$$\sum_{S: e \in \delta(S)} y_S \leq c_e, e \in E$$

We now state the primal slackness condition for this problem:
For each $e \in E, x_e \neq 0 \Rightarrow \sum_{S: e \in \delta(S)} y_S = c_e$.
The following relaxation of the dual condition would allow a 2-approximation algorithm:
$S \subseteq V, y_S \neq 0 \Rightarrow \sum_{e: e \in \delta(S)} x_e \leq 2f(S)$

This "desirable" dual condition implies that the number of picked edges crossing the cut is at most 2. However, we do not know how to ensure this condition. Despite this fact, we can still design a 2-approximation algorithm for the problem.

The primal-dual based algorithm, presented below, differs from the algorithms we have seen so far that followed this paradigm. The reason is that it has a "clean-up" step (step 3).
Define set $S \subseteq V$ to be *active* if it is a minimal set in the current iteration for which $f(S) = 1$, but there is no edge selected on the cut between S and $\overline{S}$. It follows that S is active iff it is a connected component for which $f(S) = 1$. This is true because any subset of a connected component has an edge leaving it, and is therefore not active, but any group of several connected components is not minimal, since at least one of the connected components must also be active.
The algorithm that uses this concept to produce a 2-approximation in polynomial time is as follows:

1. Initialize $F \leftarrow \emptyset$. For each $S \subseteq V, y_S \leftarrow 0$.

2. Until all sets are satisfied do: raise y_S simultaneously for each active set S, until some edge e goes tight (i.e. $y_S = c(e)$ for $e \in \delta(S)$.) It is easy to find the active sets by examining each connected component, and determining if $f(S) = 1$ for that component; that is, if there exist a vertex in the component and a vertex outside the component that need to have a path connecting them.

3. Prune the set by removing all edges that are not needed to ensure feasibility. That is, $F' = \{e \in F | F \backslash \{e\}\}$ is primal infeasible.

Remark Note that the pruning step is needed to ensure a reasonable approximation, by considering the example (see figure 1) of a star with a central node t, connected to n vertices, each with edge cost 1, and one edge, to node s, with edge cost 3. The only requirement for the problem is that s and t must be connected, so the optimal solution obviously is to just connect that one edge with cost 3. However, all of the n edges of cost 1 will become tight before the edge with cost 3, so the second step of the above algorithm will add all n edges first. Without the pruning step, this would make it an n-approximation instead of a 2-approximation.

It should be clear that the algorithm will produce a forest, because an edge will only be added if there is no connection between S and $\overline{S}$, so adding any edge cannot create a cycle.

By the weak duality theorem, $\sum_{S \subseteq V} y_S \leq \sum_{e \in F'} c_e$ for any solution, and, more specifically, for the optimal solution. To show a 2-approximation, all that needs to be demonstrated is that $\sum_{e \in F'} c_e \leq 2 \sum_{S \subseteq V} y_S$. This will be done in the next set of notes.

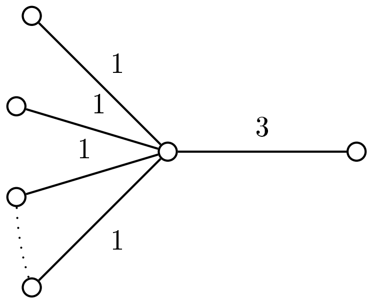

Figure 1: An example where the pruning step is needed.