CS880: Approximation Algorithms

Scribe: Matt Elder | **Lecturer:** Shuchi Chawla

Topic: Lagrangian Relaxation | **Date:** 4/26 and 4/27, 2007

We have seen many examples of the utility of linear programming. In some cases, to round an LP solution to an integer solution demands that we relax a constraint that we prefer to maintain. The Lagrangian technique will yield a method to maintain such constraints. This technique is especially useful for bicriteria optimization problems, that is, problems with two objectives where we have a fixed bound on one objective and want to optimize the other.

For example, recall the k-median problem: We are given a set of customers, a set of facilities, and a routing cost from each customer to each facility. We want to open no more than k facilities, while minimizing the total routing cost. Using standard LP techniques, it is difficult to round a relaxed LP solution to an integer LP solution without using more than k facilities. Lagrangian relaxation provides a workaround for this problem, so that we can guarantee that the final, integer solution obeys the k-facility constraint.

To demonstrate the technique of Lagrangian relaxation, we consider a solution to the k-minimum spanning tree problem. An approximation to k-median can be obtained in a similar way.

26.1 k-Minimum Spanning Trees

In an instance of the k-minimum spanning tree problem, we have a graph $G = (V, E)$, a cost $c_e > 0$ for each edge in E, and a root vertex $r \in V$. We want to find a tree that connects at least k nodes to the root while minimizing the total cost for the tree's edges. We are free to choose the set of k vertices that we will connect to the root r; call this set S.

Note that the relationship between the k-MST problem and the prize-collecting Steiner tree problem is analogous to the relationship between the k-median problem and the facility location problem. Both k-MST and k-median contain a constant-size-set restriction, which performs the task of an extra cost parameter in prize-collecting Steiner tree and facility location. As we will see, this relationship is central to the idea of the Lagrangian relaxation technique.

The integer LP for k-MST is as follows:

$$y_v = \begin{cases} 1, & v \text{ is not in the tree.} \\ 0, & \text{otherwise.} \end{cases}$$

$$x_e = \begin{cases} 1, & e \text{ is in the tree.} \\ 0, & \text{otherwise.} \end{cases}$$

$$\sum_{e \in \delta(S)} x_e \geq 1 - y_v, \ \forall S \subseteq V \setminus \{r\}, \forall v \in S.$$

$$\sum_{v \in V} y_v \leq n - k. \qquad (*)$$

$$\text{minimize} \sum_{e \in E} c_e x_e.$$

To use LP techniques, we need to relax this integer LP to a real-valued LP, and somehow still be able to respect Constraint $*$ when we round real values back to integers. We shall do this by introducing a family of LPs, parameterized by the Lagrange multiplier λ. We can think of varying λ as varying the cost of omitting vertices from our tree. It's important to note that λ is not, itself, a variable of the LP. It is a parameter of the LP, and is constant with respect to any routine that produces LP solutions. So, define the linear program LR_λ as follows:

$$y_v \in [0, 1]$$

$$x_e \in [0, 1]$$

$$\sum_{e \in \delta(S)} x_e \geq 1 - y_v, \ \forall S \subseteq V \setminus \{r\}, \forall v \in S.$$

$$\text{minimize} \sum_{e \in E} c_e x_e + \lambda \left(\sum_v y_v - (n - k) \right).$$

The term $\lambda \left(\sum_v y_v - (n - k) \right)$, above, replaces Constraint $*$ in the original LP. For any λ, the LP LR_λ has the same optimal solution as the following prize-collecting Steiner tree LP, PCST_λ:

$$y_v \in [0, 1]$$

$$x_e \in [0, 1]$$

$$\sum_{e \in \delta(S)} x_e \geq 1 - y_v, \ \forall S \subseteq V \setminus \{r\}, \forall v \in S.$$

$$\text{minimize} \sum_{e \in E} c_e x_e + \lambda \left(\sum_v y_v \right).$$

Allowing LP names to stand for the optimal values of their objective functions, it's clear that $\text{PCST}_\lambda - \lambda(n-k) = \text{LR}_\lambda$. Furthermore, any solution to the k-MST problem is a feasible solution to LR_λ; when Constraint $*$ is tight, as it is for all solutions to the k-MST problem, then the objective value of this solution is the same in both problems. So, $\text{LR}_\lambda \leq \text{OPT}$. (OPT is the optimal solution to k-MST. Remember, that's the problem that we're (still) trying to approximate.)

Let PCST'_λ be the integer solution to PCST_λ yielded by the LP-dual algorithm. If we let $\lambda = 0$, then PCST'_λ is a tree containing only the root because there is no penalty for leaving unused vertices. Similarly, if we let $\lambda = \max_e c_e$, then PCST'_λ will contain all vertices because the penalty for unused vertices dominates the cost of expanding the tree. So, it seems like there should be some moderate value of λ for which PCST'_λ contains nearly k vertices. This need not quite be the case, but we can use binary search to find two values of lambda, $\lambda_1 \approx \lambda_2$, for which we get two trees T_1 and T_2 such that $|T_1| < k < |T_2|$. From these trees, we can interpolate a solution using exactly k vertices. However, with luck, this interpolation may not be necessary.

Theorem 26.1.1 *If $PCST'_\lambda$ has k vertices, then it gives a 2-approximation to k-MST.*

Proof: Let $x, y \stackrel{\text{def}}{=} \text{PCST}'_\lambda$. Since PCST'_λ has k vertices, we know that $\sum_v y_v = n - k$. Then, by our analysis of Problem 4 in Homework 3,

$$\sum_e c_e x_e + 2\lambda \sum_v y_v \leq 2\text{PCST}_\lambda, \text{ so}$$

$$\sum_e c_e x_e \leq 2\left(\text{PCST}_\lambda - \lambda \sum_v y_v\right) = 2\left(\text{PCST}_\lambda - \lambda(n-k)\right) = 2\text{LR}_\lambda \leq 2\text{OPT}.$$

■

If we are unable to find a λ such that PCST'_λ has exactly k vertices, then we need to find a way to combine T_1 and T_2 into a single tree, which does not cost much more than OPT. Let $\lambda_1 = \lambda_2$; except that they generate two different trees, we assume that the difference between λ_1 and λ_2 is negligible.

Let μ_1 and $\mu_2 \stackrel{\text{def}}{=} 1 - \mu_1$ satisfy $\mu_1 k_1 + \mu_2 k_2 = k$, where $k_1 = |T_1|$ and $k_2 = |T_2|$. Then:

$$\mu_1 = \frac{k_2 - k}{k_2 - k_1}$$
$$\mu_2 = \frac{k - k_1}{k_2 - k_1}$$

Now, letting $c(T)$ denote the cost of tree T, we know the following

$$\begin{aligned} c(T_1) + 2\lambda(n - k_1) &\leq 2\text{PCST}_\lambda, \text{ and} \\ c(T_2) + 2\lambda(n - k_2) &\leq 2\text{PCST}_\lambda, \text{ so} \\ \mu_1 c(T_1) + \mu_2 c(T_2) + 2\lambda(n - \mu_1 k_1 - \mu_2 k_2) &\leq 2\text{PCST}_\lambda, \text{ which yields} \\ \mu_1 c(T_1) + \mu_2 c(T_2) \leq 2\left(\text{PCST}_\lambda - \lambda(n-k)\right) &\leq 2\text{OPT}. \end{aligned}$$

If $\mu_2 \geq \frac{1}{2}$, then $c(T_2) \leq 2\mu_2 c(T_2) \leq 4\text{OPT}$. Since $|T_2| > k$, we can simply use T_2 as our solution.

Otherwise, $\mu_1 \geq \frac{1}{2}$. Let $T_2' \stackrel{\text{def}}{=} T_2 \setminus T_1$. The following subroutine, Find-Subtree, will find a subtree of T_2 of size at least $(k - k_1)$.

Find-Subtree:

1. Exchange each undirected edge of T_2 for two directed edges of the same cost, one pointing each way. These edges form an Euler tour containing all vertices of T_2'. Note that each vertex appears twice in the tour.

2. From each vertex in T_2', start following the Euler tour in a clockwise direction until $2(k - k_1)$ nodes of T_2' are encountered, including repeats. This gives us at least $2(k_2 - k_1)$ different subpaths of the Euler tour, two for each vertex in T_2'.

3. Return the shortest such subtour.

Each edge of the Euler tour belongs to exactly $2(k - k_1)$ subpaths and there are at least $2(k_2 - k_1)$ subpaths in all. Therefore, since the cost of the entire Euler tour is $2c(T_2)$, one of the subpaths has length at most $\frac{2(k-k_1)}{2(k_2-k_1)} 2c(T_2)$.

So, suppose that Find-Subtree outputs the tree S. S contains at least $(k - k_1)$ distinct nodes of T_2, and costs at most $\frac{2(k-k_1)}{k_2-k_1} c(T_2) = 2\mu_2 c(T_2)$.

We build the interpolated tree by starting with T_1, adding S, and adding the shortest path from T_1 to S. The first piece has cost $c(T_1)$ and the second has cost $c(S) \leq 2\mu_2 c(T_2)$. If we have preprocessed the graph to throw away all nodes whose distance to the root is greater than OPT, we can ensure that this last path has cost no more than OPT. We don't know what OPT is, so we'll need to run this entire algorithm n times; on run i we remove the i vertices farthest from the root.

Thus:

$$\begin{aligned} \text{total cost} &= c(T_1) + c(S) + \text{ cost of shortest path} && (26.1.1)\\ &\leq 2\mu_1 c(T_1) + 2\mu_2 c(T_2) + \text{OPT} && (26.1.2)\\ &\leq 4\text{OPT} + \text{OPT} && (26.1.3)\\ &= 5\text{OPT}. && (26.1.4) \end{aligned}$$

Thus, the technique of Lagrangian relaxation gives us this algorithm, a 5-approximation to the k-minimum spanning tree problem.