22C:253

Scribe: George Thomas

November 14, 2002

A factor-f algorithm for SET COVER via the primal-dual framework
The primal of this problem, which is the LP-relaxation for SET COVER is the following:

Minimize

$$\sum_{j=1}^{n} x_j \cdot c(S_j)$$

subject to

$$\begin{aligned} \sum_{j: i \in S_j} x_j &\geq 1 \text{ for each } i = 1, 2, \cdots, m \\ x_j &\geq 0 \text{ for each } j = 1, 2, \cdots, n \end{aligned}$$

The dual of this problem is:
Maximize

$$\sum_{i=1}^{m} y_i$$

subject to

$$\begin{aligned} \sum_{i \in S_j} y_i &\leq c(S_j) \text{ for each } j = 1, 2, \cdots, n \\ y_i &\geq 0 \text{ for each } i = 1, 2, \cdots, m \end{aligned}$$

The *primal complementary slackness* condition is:
For each $j = 1, 2, \cdots, n : x_j = 0$ or $\sum_{i \in S_j} y_i = c(S_j)$
The *dual complementary slackness* condition is:
For each $i = 1, 2, \cdots, m : y_i = 0$ or $\sum_{j: i \in S_j} x_j = 1$
The corresponding *approximate primal complementary slackness* condition is:

$$\frac{c(S_j)}{\alpha} \leq \sum_{i \in S_j} y_i \leq c(S_j)$$

The corresponding *approximate dual complementary slackness* condition is:

$$\beta \geq \sum_{j: i \in S_j} x_j \geq 1$$

(Note that $\alpha = 1$ and $\beta = f$ gets us the original "exact" constraints.)
We would like these two approximate constraints to be maintained. If we can produce x and y such that x is a feasible, integral, primal solution and y is a feasible dual solution satisfying these approximate constraints, then x is a factor-f approximation solution for SET COVER.

Remarks on the approximate constraints
Approximate Dual Constraint:
How hard is it to maintain the dual constraint? Easy. (It comes for free and is always satisfied.) If x is a feasible integral solution, then the approximate dual complementary slackness condition is satisfied.
Approximate Primal Constraint:
Another way to write this condition is as follows:
For each $j = 1, 2, \cdots, n : x_j \neq 0 \Rightarrow \sum_{i \in S_j} y_i = c(S_j)$
This suggests a way of setting x_j's to 1's: when a set S_j becomes "tight" (ie. $\sum_{i \in S_j} y_i = c(S_j)$) then set the corresponding $x_j = 1$

Algorithm

1. Set $x = 0$ (integral, infeasible primal solution) and $y = 0$ (feasible dual solution).
   Note that approximate primal complementary slackness condition is satisfied. The approximate dual complementary slackness condition is NOT satisfied after the initial step, after we start increasing y_i's. We do not worry about this because as soon as x becomes feasible, the dual constraint will be re-satisfied.

2. Pick an uncovered element i. Increase y_i to the minimum value such that some set containing i is tight.

3. For all sets S_j that are tight, set $x_j = 1$ (ie. throw set S_j into solution).

4. Remove all elements i covered by sets in solution. (This simply means their current y_i's cannot be increased any further.) Go back to step 2.

In step 2, suppose we increased y_i's "synchronously". The first tight set is a set S_j with minimum $\frac{cost(S_j)}{|S_j|}$. This is equivalent to our greedy choice in the greedy algorithm approach.

Steiner Tree
Input: A graph $G = (V, E)$ with edge costs $C : E \to Q^+$ and a set $R \subseteq V$ of *required* vertices.
Output: A tree in G with minimum cost containing R. (If R is a tree, this becomes the minimum spanning tree problem which we know how to solve in P. The generalization is in NP. The hard part comes from choosing which vertices not in R should participate in the solution.)
Status: Easy factor-2 approximation algorithm via minimum spanning tree. The approximation factor has been improved many times in the last decade (eg. 5/3-factor, all subsequent lower factors are due to Zelikovsky).

There is a specific version of this problem called *Euclidean Steiner Tree*
Input: Points in $\Re^n$
Output: A tree with smallest cost connecting these points, but may include other points as well.

Status: There is a PTAS for this (due to S. Arora).

Solving the Steiner Tree problem

1. Reduce the problem to the Metric Steiner tree problem.
   Specifically, construct the following:

   $$G = (V, E) \rightarrow G^M = (V, E^M)$$

   where G^M is the complete graph and $c(u, v)$ = cost of cheapest path between u and v in G.

2. Solve the Steiner Tree problem on G^M and R. This is called the *Metric Steiner Tree* problem. (The edge costs of G^M satisfy triangle inequality.)

   Lemma 1 *Cost of optimal Steiner tree of R in G = cost of optimal Steiner tree of R in G^M.*

   Proof: This is clear from the fact that for any edge (u, v), its cost in G^M is no more than its cost in G. So we might as well solve the problem on G^M. □

3. Compute a minimum spanning tree T of G^M.

   Lemma 2 *Cost of T, cost$(T) \leq 2 \cdot OPT$*

   Proof: Consider an inorder traversal or tour of the edges in the optimal Steiner tree. When backtracking, we skip the vertices that we have already traversed by adding an edge to an unvisited vertex in our graph. So it is clear that we can, at most, end up doubling our edges. These shortcut edges do not increase the cost of the tour because of triangle inequality. Hence, the cost of our tour $\leq 2 \cdot OPT$. Remove one edge in this cycle and we get a path $\leq 2 \cdot OPT$. So if we used the minimum spanning tree, it would have to be less than this. □

Steiner Forest
The algorithm we describe is by Goemans and Williamson (factor-2 approximation, best known).
Input: A graph $G = (V, E)$ with edge costs $C : E \rightarrow Q^+$. A collection of subsets of V, $S_1, S_2, \cdots, S_k$
Output: A subgraph of G with minimum cost such that for any S_i, vertices in S_i lie in the same connected component of the subgraph.