CSE5401 Lec. 20

Primal-Dual Algorithms II: Steiner Forest

1. Steiner Forest problem
2. Primal-dual algorithm: increasing many variables at once
3. Analysis

So far, we have seen a couple of examples of algorithms using the primal-dual method. These algorithms all had the form

1. Start with a simple feasible solution to the dual LP of a LP relaxation of some integer optimization problem
2. Choose some "minimal" dual variable (e.g., min. size set) for which the associated constraints are not tight.
3. Increase the value of the variable until some constraints become tight, and add one of these constraints' primal variables to the chosen set.

(Repeat 2-3 until all dual variables have a tight constraint)

Today we'll see a different style of primal-dual analysis, where instead of increasing one dual variable at a time, we increase many simultaneously, and at the end bound the average number of times a set is hit, rather than the maximum (We'll also do a final "clean up" of the solution we construct.)

We'll give an algorithm for the steiner forest problem: Given an undirected graph $G=(V,E)$ with costs $c_e \geq 0$ for $e \in E$ and a list of k pairs $(s_1,t_1), \ldots, (s_k,t_k) \in V \times V$, find a subgraph $F \subseteq G$ s.t. each s_i, t_i are connected in F,

minimizing the total cost $\sum_{e \in F} c_e$.

Note: steiner forest generalizes minimum spanning tree (all pairs must be connected) and shortest path (one pair)

As with these problems, optimal solutions to steiner forest do not contain cycles (we can delete an edge from a cycle, reducing the cost, while preserving connectivity) hence the solution is a forest (connected components are trees). We'll give an algorithm that essentially generalizes Kruskal's algorithm for minimum spanning tree. (But unlike minimum spanning tree, steiner forest is NP-complete so we will only get an approximately optimal forest.)

Let's first write a linear program formulation. Naturally we'd like variables x_e for each $e \in E$, $x_e \geq 0$, s.t. $F = \{e : x_e \geq 0\}$

Then the cost is $\sum_{e \in E} c_e x_e$.

It is less obvious how to write the constraint that s_i & t_i are connected in F. but we take inspiration from the connection between flows & cuts: observe that a path from s_i to t_i is a flow from s_i to t_i in G_i this flow must contain an edge from every $s_i - t_i$ cut in G. Indeed, if not, then for the cut S^* with $s_i \in S^*$, $t_i \notin S^*$, none of the vertices outside S^* is reachable in F by hypothesis, and therefore in particular, t_i is not reachable from s_i. Therefore, a feasible F must satisfy the cut constraints

$$\sum_{e \in S} x_e \geq 1 \quad \text{for all } s_i - t_i \text{ cuts } S, \; i = 1, \ldots, k$$

Conversely, if t_i is not reachable from s_i, then the set of vertices

S reachable from s_i gives a cut S s.t. $t_i \notin S$, and $\sum_{e \in S} x_e = 0$. Thus, the cut constraints are also sufficient to capture that all pairs (s_i, t_i) are connected in F.

What is the dual linear program?

We have a dual variable w_S for every s_i-t_i cut for $i = 1, \dots, k$.

We maximize $\sum_{s.t.\ cut\ S} w_S$ s.t. $\sum_{cuts\ S : e \in S} w_S \leq c_e$ for all $e \in E$, $w_S \geq 0$.

Now, as we turn to the algorithm, it turns out that unfortunately, we might always have to consider a large cut, so simply counting the number of edges is not sufficient. On the other hand, in any forest the number of edges is always at most #vertices - 1. We will be able to use this relationship to bound the average (hence, total overall) number of times the cuts are crossed.

To utilize this aggregate relationship, we don't simply maximize the dual variables one at a time, but rather increase several of them simultaneously until one of the edge constraints — the "shortest" (least-costly) one — becomes tight. We add this edge to the forest and resume. The variables we increase correspond to the connected components of the current forest — so if we ignored the contribution to the dual variables from prior iterations, we'd have Kruskal's algorithm. Finally, after all of the pairs are connected, we iterate over the set of chosen edges and delete any that form cycles.

In summary, here is the algorithm (let $\partial S \stackrel{def}{=} E \cap S \times (V - S)$)

Initialize $\vec{w} = \vec{0}$, $F_0 = \emptyset$, $i = 0$

While some pair (s_j, t_j) is not connected in F_i

  Increment i.

Put $\mathcal{C}_i \leftarrow \{C \subseteq G : C$ is a connected component in F_{i-1} with exactly one of s_j or t_j for some $j\}$

Increase all w_C for $C \in \mathcal{C}_i$ by same amount until some $e_i \in \partial C$ has $\sum_{S : e_i \in \partial S} w_S = c_{e_i}$ ("e_i is tight")

Put $F_i \leftarrow F_{i-1} \cup \{e_i\}$

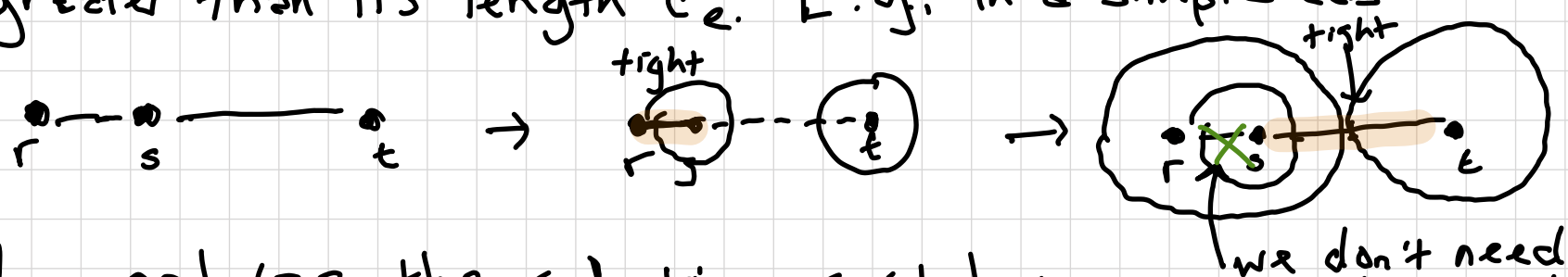
Put $F' \leftarrow F_i$

For $j = 1, \ldots, l$

| If $F' - e_j$ connects all pairs put $F' \leftarrow F' - e_j$

Return F'

We think of w_C as a "moat" around C, with the constraint that the total width of the moats under an edge e cannot be greater than its length c_e. E.g. in a simple case



We analyze the solution cost by bounding the total increments across iterations.

Observe: only the edges of F' in ∂C for $C \in \mathcal{C}_i$ contribute to the final cost, where if we shrink each $C \in \mathcal{C}_i$ to a single vertex, this is a forest (since F' has no cycles) on $|\mathcal{C}_i|$ vertices

Lemma $\sum_{C \in \mathcal{C}_i} |F' \cap \partial C| \leq 2|\mathcal{C}_i|$

Proof: Recall that a forest on n vertices has at most $n-1$ edges, where the edges between $C \neq C' \in \mathcal{C}_i$ are each counted twice in the sum. We only need to bound the contribution of edges from some $\mathcal{C}_i$ (containing exactly one of s_j or t_j for some j) to another vertex in G. Since

the edge is in F'. it must be on the (shortest) path from some s_j to some t_j - meaning, the connected component in F_i cannot contain both and must not contain one, or else it would be in $C_{i,1}$ so it contains neither, and hence must contain another edge exiting it in F'.

Thus, if we let $\tilde{C}_i$ denote the connected components of F_i not in $C_{i,1}$ $\sum_{C \in \tilde{C}_i} |\delta C \cap F'| \leq 2|C_i| + 2|\tilde{C}_i|$ and $\sum_{C \in \tilde{C}_i} |\delta C \cap F'| \geq 2|\tilde{C}_i|$ so

$$\sum_{C \in C_i} |\delta C \cap F'| = \sum_{C \in C_{i,1} \cup \tilde{C}_i} |\delta C \cap F'| - \sum_{C \in \tilde{C}_i} |\delta C \cap F'| \leq 2|C_i| + 2|\tilde{C}_i| - 2|\tilde{C}_i| \quad \square$$

Theorem The algorithm returns a steiner forest for the given pairs of cost at most twice the minimum

Proof: F' consists of tight edges in the final feasible $\vec{w}$ so $\sum_{e \in F'} c_e = \sum_{e \in F} \sum_{S : e \in \delta S} w_S = \sum_{\text{cut } S} |F' \cap \delta S| w_S$. We prove by induction on i that on every ith iteration $\sum_{\text{cut } S} |F' \cap \delta S| w_S \leq 2 \sum_{\text{cut } S} w_S$: initially $w_S = 0$ for all S, and in iteration i, we only increase the value of w_C for $C \in C_i$ by the same amount ε_i. This increases LHS by $\varepsilon_i \cdot \sum_{C \in C_i} |F' \cap \delta C| \leq \varepsilon_i \cdot 2|C_i| = 2 \sum_{C \in C_i} \varepsilon_i$ by the lemma, where this is the amount the RHS increases by. This completes the induction step. By weak duality, $\sum_{\text{cut } S} w_S \leq \sum_{e \in E} c_e x_e^*$ for an optimal fractional $\vec{x}^*$, which is also less than the minimum cost of a Steiner forest (integer solution). Thus, indeed, $\sum_{e \in F'} c_e \leq 2 \sum_{e \in F^*} c_e$ for an optimal F^*. $\square$