COMPSCI 590: Graph Algorithms March 22, 2015

Lecture 18

Lecturer: Debmalya Panigrahi *Scribe: Nat Kell*

1 Introduction

In this lecture, we examine the Steiner tree and forest problems in the *online* setting, i.e., terminals are revealed to algorithm in an unknown sequence and connectivity must be maintained after each terminal arrival. We will examine two algorithms: the first is the natural greedy algorithm (connect each terminal using cheapest extension of the current solution), which we show achieves competitive ratios $O(\log k)$ and $O(\log^2 k)$ for the Steiner tree and forest cases, respectively, where k is the number of terminals. We will then show how to slightly modify this greedy algorithm to improve this bound to $O(\log k)$ for the Steiner forrest case (which is optimal up to constants).

2 Online Steiner Tree

2.1 Online algorithms

In all the problems we have previously examined this semester, we assumed the entire input for the problem is available at the outset of the algorithm. However this assumption can be unrealistic; many time in practice, the input for a problem arrives over time and decisions must be made before seeing the entire input. This motivates examining problems under the *online model*. In the online paradigm, each piece of input is revealed to the algorithm in an unknown sequence and the algorithm must make irrevocable assignment(s) for the current piece of input before receiving the rest of the sequence.

Since the algorithm must make decisions before seeing the entire input, some of its assignment are likely to be suboptimal. However, much like the analysis we have done for approximation algorithms for NP-hard problems, we would like to be able bound this suboptimality and claim that the algorithm's solution is never too far from the best possible offline solution. Therefore, we define the *competitive ratio* of an online algorithm to be the worst case ratio, over all input sequences I, between the cost the online algorithm on I to the cost of the optimal solution for I. Note that this is exactly the same definition as the approximation ratio for offline problems; the only difference is that here, the algorithm's suboptimal performance is due to a lack of future information (whereas in the offline setting, suboptimality is forced by restricting the amount of time we have for solving the problem).

2.2 Online Steiner tree definition

In the online setting, the Steiner tree problem is defined as follows. At the outset, the algorithm is given an *entire* graph $G = (V, E)$ with non-negative edge weights. Terminal vertices $\langle t_1, \ldots, t_k \rangle = T \subseteq V$ are then revealed to the algorithm one at a time in an unknown sequence. When a terminal t_i arrives, the algorithm is required buy additional edges (i.e. add these edges to its current solution) so that for each previously revealed terminal t_j, there is some path from t_i to t_j in the algorithm's solution. Like in the offline problem, the goal is to minimize the total cost of the edges bought by the algorithm.

2.3 The Greedy Algorithm

One can show the following lower bound for the online Steiner tree (although we will not cover its proof in this lecture):

Theorem 1. *The competitive ratio of any online algorithm for Steiner Tree is* $\Omega(\log k)$*, where k is the number of terminals.*

The most natural algorithm to try first is the following greedy approach: For each terminal arrival t_i, we buy the cheapest set of unpurchased edges that connects t_i to the algorrthm's current solution, i.e., the shortest shortest path between t_i and all previous terminals t_j where $j < i$. For the Steiner tree problem, this turns out to be optimal (however, later on we will need a knew approach in order to solve the Steiner forest problem optimally). For the rest of the section, it will be our goal to show the following:

Theorem 2. *The greedy algorithm for Steiner tree has competitive ratio* $O(\log k)$*, where k is the number of terminals.*

Therefore this result, along with Theorem 1, shows that the greedy algorithm is optimal for online Steiner tree.

2.4 Greedy algorithm analysis via dual fitting

To show Theorem 2, we will use a technique known as *dual fitting*. The overall approach will be to relate the cost of the algorithm's (primal) solution to a feasible set of dual solutions. By weak duality, we know that the cost of any feasible dual solution is a lower bound on the optimal primal cost; therefore, if we bound the cost between the primal and these dual solutions, then we establish competitiveness. Note that in this case, the online algorithm *does not* know about or work with these dual solutions in any way—the dual structure is only being exploited for our analysis in hindsight.

More formally, suppose that for any run of the algorithm, we can construct β feasible dual solutions (i.e., each dual solution is a complete and independent assignment of the variables in the dual LP) such that the final primal cost is at most γ times the total cost of all β dual objectives. Since each dual is feasible, the average dual solution D' must be feasible, as well; therefore, the algorithm's cost is at most $\gamma \cdot \beta$ times the cost of D', which by weak duality (as discussed above) implies that the algorithm is $(\gamma\beta)$-competitive. For our analysis in this section, we set $\gamma = O(1)$ and $\beta = \log k$, which implies the desired $O(\log k)$ competitive-ratio bound.

For completeness, recall the primal LP for Steiner tree (relaxation):

$$\begin{aligned} &\min \sum_{e \in E} c_e x_e \\ &s.t. \ \forall\, S \in \mathcal{T} \sum_{e \in \delta(S)} x_e \geq 1 \\ &x_e \geq 0, \end{aligned}$$

where c_e is the weight of edge e in G, $\mathcal{T}$ is the set of cuts on G that separate at least one pair of terminals, and $\delta(S)$ is the set of all edges crossing cut S. The dual of this LP is given as follows:

$$\max \sum_{S \in \mathcal{T}} y_S$$
$$s.t.\ \forall e \in E \sum_{S: e \in \delta(S)} y_S \leq c_e$$
$$y_S \geq 0.$$

By scaling we will assume the cost of the optimal Steiner tree is k. Therefore, we will also assume that all shortest paths picked by the algorithm have length at least 1. This is a safe assumption since the total cost of paths with length less than 1 is at most k, and therefore the cost added from these paths can be charged to the optimal solution without affecting our bound asymptotically.

With these assumptions, we partition the terminals into $\log k$ classes $C_1, \ldots, C_{\log k}$ based on the lengths of the shortest paths chosen to connect them to the current Steiner tree. Namely, we place terminal t_i with shortest-path length ℓ_i in class C_j if $2^{j-1} < \ell_i \leq 2^j$. We will associate a dual LP D_j with each class C_j, i.e., we will add cost to dual D_j for all terminals in class C_j. Similar to the analysis we used for offline Steiner tree, we will think of a dual update for terminal t_i as placing a ball centered at vertex t_i. Here we will choose the radius of this ball to be 2^{j-2} if t_i is in class j (i.e., we will grow a ball of radius 2^{j-2} starting at set the $\{t_i\}$; see the lectures notes for offline Steiner tree for more explicit details on how exactly this ball growing process is implemented in terms of setting dual variables).

Since the primal increase for a terminal in class C_j is at most 2^j and the added cost to D_j is 2^{j-2}, the primal increase is within a constant factor of the corresponding dual increase (i.e. we can set $\gamma = 4 = O(1)$). Thus, it remains to be shown that each dual is feasible after we run this process. Namely, we will show that we do not place any "overlapping" balls in a given dual D_j (which is enough to show dual feasibility). For sake of contradiction, assume that at least two balls do in fact overlap in dual D_j. Since the balls in dual D_j have radius 2^{j-2}, this implies there exists two terminals t_i and $t_{i'}$ that are at most $2^{j-2} + 2^{j-2} = 2^{j-1}$ distance from one another where $i < i'$. However, we placed terminal $t_{i'}$ in class C_j because its distance to the closest previous terminal was greater than 2^{j-1}, which is a contradiction since terminal t_i is at most distance 2^{j-1} from $t_{i'}$.

Therefore, each dual D_j we construct is feasible. By our earlier discussion, this implies Theorem 2, i.e., the greedy algorithm is $O(\log k)$ competitive for online Steiner tree.

3 Online Steiner Forest

We will now examine the Steiner *forest* problem in the online setting. Recall that in the Steiner forest problem, we are again given an undirected edge weighted graph G, but instead of a sequence of terminals, we are given a sequence of terminal pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. Now the goal is to find a subgraph F of minimum cost such that there exists a path between terminals s_i and t_i in F for all terminal pairs (s_i, t_i). In the online setting, pairs are revealed in a sequence, and connectivity must be established between an arriving pair before observing future terminal pairs.

3.1 Greedy algorithm for Steiner forest

Since the greedy algorithm proved to be competitively optimal in the Steiner tree case, it seems like it should be a first good attempt at an algorithm for Steiner forest. In order to make the approach viable, we will use a slightly alternate definition of greediness. When a terminal pair (s_i, t_i), we will connect them using the shortest path between them in the current graph; however now, we will then contract all the edges along this path into a single node before continuing (another way of defining the algorithm is that to say that we are picking the shortest path such that we only pay for the edges we have not selected yet).

In the Section 3.2 , we will show the following theorem.

Theorem 3. *The greedy algorithm for Steiner tree has competitive ratio $O(\log^2 k)$, where k is the number of terminals.*

Observe that we will have to lose an extra $\log k$ factor in order for our analysis to work. Later we will outline how to refine the greedy algorithm in order to bring the bound back down to $O(\log k)$. Note that there are no known lower bounds showing that greedy algorithm is $\omega(\log k)$ competitive; however as of now, our best analysis gives a $O(\log^2 k)$ competitive ratio.

3.2 Greedy algorithm analysis via dual fitting and girth theorems

We will run into issues if we try to directly employ our dual fitting analysis from the Steiner tree case. Naively, if the shortest path between s_i and t_i is length 2^j, we would just place balls of radius 2^{j-3} around both s_i and t_i in D_j (using the same dual/class structure we used before). Recall that the way we arrived a contradiction in the Steiner tree analysis is that within a given dual D_j if we had overlapping dual balls for two terminals t_i and $t_{i'}$, it implied that $t_{i'}$ was not assigned greedily since connecting to t_i was an available choice and such an overlap for implies t_i and $t_{i'}$ are too close to be placed in dual D_j. But in the Steiner forest case, we may not be required to connect $t_{i'}$ to t_i, and thus overlapping dual balls do not necessarily contradict the greediness of the algorithm.

We will fix this issue by still trying our best to stick to original analysis, while essentially "skipping" ball placements that will create an overlap. We will then use a separate charging argument to establish that the cost that we do not account from these skips can be bounded by the cost that we charge to actual ball placements. We will also lower our expectations a bit by reducing the radii of the balls by a $\log k$ factor.

More explicitly, we will place attempt to place dual balls around both s_i and t_i of size $2^{j-3}/\log k$ in dual D_j (where the shortest path between s_i and t_i is between 2^{j-1} and 2^j). If either ball can be placed without overlapping any previously placed dual balls, then we go ahead and do the placement. Otherwise, there exists previous terminals a and b such that the balls around s_i and t_i overlap with the balls around a and b, respectively. Since we are not placing a ball around s_i nor t_i, we will instead add a *meta edge* between terminals a and b. We will call the resulting graph formed by meta edges in dual D_j the *meta graph* M_j.

Our first step in this new charging scheme is to show the following lemma.

Lemma 4. *If for all $j = 1, \ldots, \log k$, meta graph $M_j = (V_j, E_j)$ has $O(k_j)$ edges where $|V_j| = k_j$, then the greedy algorithm is $O(\log^2 k)$ competitive.*

Proof. Based on the construction, each vertex in M_j is a terminal that we successfully placed around in dual D_j. Each edge we place in M_j corresponds to two dual balls that we could not place in D_j; however since there are $O(k_j)$ edges in M_j, it follows that the total cost from these skipped ball placements must be within a constant factor of the to cost of the balls we do place since each ball in D_j is of equal size.

Therefore, the total cost of our dual solution is $O(\log k)$ times the total primal cost, since the balls we are placing are scaled down by a $\log k$ factor. Since there are $\log k$ dual solutions, it follows that the algorithm is $O(\log^2 k)$ competitive (we are setting $\gamma = \beta = \log k$; see the dual fitting parameters we discussed earlier in Section 2.3). □

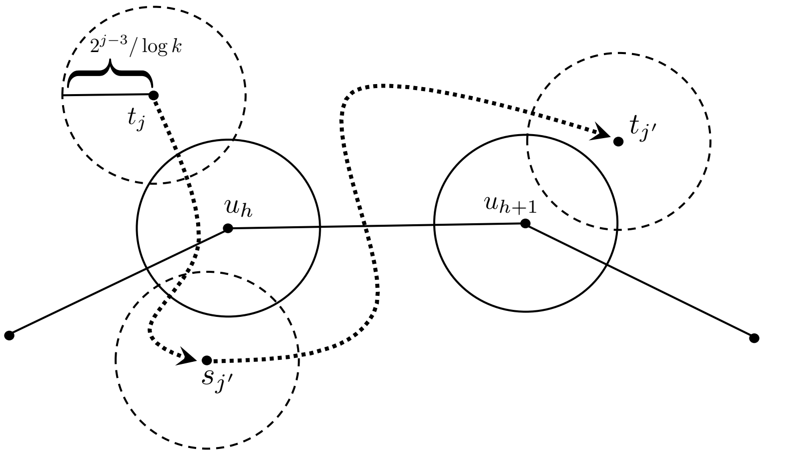

Figure 1: Illustration of the argument in Lemma 7. Since both the balls around t_j and $s_{j'}$ overlap with u_h's ball, there must be a path from t_j to $s_{j'}$ of length at most $2^{j-1}/\log k$ (the first dotted path). Since $s_{j'}$ and $t_{j'}$ must be connected in the algorithm's solution, the second dotted path can be used for free (given our modified notion of greediness).

Therefore in order to establish Theorem 3, we are left with showing that meta graph M_j has $O(k_j)$ edges. To do this, define the *girth* of a graph to be the length of its shortest cycle (in terms of the number of edges). We will utilize the following well-known theorem in extremal graph theory [1]:

Theorem 5. *For any graph $G = (V,E)$ with girth g and $|V| = n$, we have $|E| = O(n^{1+c/g})$ for some universal constant c.*

This gives us the following corollary.

Corollary 6. *If the girth of a graph $g = O(\log n)$, then $|E| = O(n)$.*

Thus, it is sufficient to establish that there cannot be a cycle shorter than length $\log k_j$ in M_j.

Lemma 7. *For $j = 1,\dots,\log k_j$, all cycles in M_j have more than $\log k_j$ edges.*

Proof. For sake of contradiction, assume that at some point during the algorithm, the arrival of terminal pair (s_i,t_i) makes us place a meta edge between previous terminals a and b such that edge (a,b) now completes a cycle of length $\ell < \log k_j$ in meta graph M_j. Let $C = \langle a = u_1,\dots u_{\ell+1} = b\rangle$ denote the sequence of terminals in this cycle. For each edge (u_h,u_{h+1}) in C, we know that there is some terminal t_j that overlapped with u_h's ball and another terminal pair $(s_{j'},t_{j'})$ where $s_{j'}$'s ball overlapped with u_h's ball and $t_{j'}$'s ball overlapped with u_{h+1}'s ball. Observe that since $(s_{j'},t_{j'})$ arrived before (s_i,t_i) in the online sequence and we contract paths that are already connected in the algorithm's solution, this gives us a path of cost at most $2^{j-1}/\log k$ from t_j to $t_{j'}$. This is true since the overlapping balls between t_j, u_h, and $s_{j'}$ imply that taking the shortest path from t_j to u_h and then taking the shortest path from u_h to $t_{j'}$ has cost at most $2^{j-3}/\log k + 2\cdot 2^{j-2}/\log k + 2^{j-3}/\log k = 2^{j-1}/\log k$ (which then allows us to take the connected path from $s_{j'}$ to $t_{j'}$ for free; see Figure 1).

Applying this argument for every meta edge in the cycle, we obtain a path of length $\ell \cdot 2^{j-1}/\log k \le 2^{j-1}$ since $\ell \le \log k_j \le \log k$; however, this is a contradiction since this implies that the shortest path between s_i and t_i is at most 2^{j-1}, and therefore we should not be placing a ball in dual D_j. □

Lemma 4, Corollary 6, and Lemma 7 collectively imply Theorem 3, i.e., the greedy algorithm is $O(\log^2 k)$ competitive for the online Steiner forest problem.

3.3 Obtaining a $O(\log k)$ competitive algorithm: Augmented Greedy

To finish the section, we will outline how to improve the greedy algorithm from Section 3.2 to obtain an $O(\log k)$ competitive algorithm. The idea is that the algorithm will still run the greedy algorithm, but at the same time maintain state regarding the dual fitting analysis and when we must place meta edges. We will we will now place balls of radius 2^{j-3}, i.e., we will not scale our ball radii down by a $\log k$ factor as we did for the vanilla greedy algorithm; however, anytime we add a meta edge between two terminals a and b because the dual balls of arriving terminals s_i and t_i intersected a and b's balls, respectively, we will then add both the shortest paths from a to s_i and from b to t_i to our solution (in addition to the greedy path between s_i and t_i).

Observe that since the shortest path between s_i and t_i is at most 2^j and intersections between s_i and a's balls and t_i and b's balls imply both pairs are at most 2^{j-2} distance from one another, these "augmented" paths will increase the algorithm's cost by at most a constant factor. The inclusion of the augmented paths will then allow us to claim that our meta graph is a *tree*, which we argue in the following lemma. This is sufficient for the dual fitting analysis from the previous section to go through while setting $\gamma = O(1)$ and $\beta = \log k$, yielding the desired $O(\log k)$ competitive ratio bound.

Lemma 8. *For $j = 1, \ldots, \log k_j$, meta graph M_j is a tree for the augmented greedy algorithm.*

Proof. We will use the same setup that we used in Lemma 7. Assume for sake of contradiction that the arrival of terminal pair (s_i, t_i) results in the placement of edge (a, b) in meta graph M_j that completes a cycle $C = \langle a = u_1, \ldots, u_\ell = b \rangle$. As discussed above, since a and s_i's balls overlap, there exists a path between a and s_i of length at most 2^{j-2}; the same argument applies for b and t_i.

Now, consider an edge (u_h, u_{h+1}) in C where $h \neq 1$ and $h + 1 \neq \ell$. Just as in the case of the vanilla greedy algorithm, it follows that there is a terminal t_j whose ball intersects u_h's ball and a terminal pair $(s_{j'}, t_{j'})$ where $s_{j'}$'s ball intersects u_h's ball and $t_{j'}$'s ball intersects u_{h+1}'s ball. However, when terminal t_j caused the placement of a meta edge (namely, (e_{h-1}, e_h)), the algorithm added a path from t_j to u_h to its solution. The same can be said for $s_{j'}$ and u_h; therefore, there exists a path in the algorithm's solution from t_j to $s_{j'}$. Clearly there is a path between $s_{j'}$ and $t_{j'}$ in the algorithm's solution, and therefore there is a path from t_j to $t_{j'}$ that is of zero cost to the algorithm.

Applying this argument for every meta edge (u_h, u_{h+1}), it follows the there exists a path of cost 2^{j-1} from s_i to t_i; this path is composed of the initial and ending paths from s_i to a and b to t_i, along with a free path from a to b via the free paths for each meta edge (whose existence we just argued). This again gives us the desired contradiction, since the shortest path for any terminal pair in dual D_j must be more than 2^{j-1}. □

References

[1] Béla Bollobás. *Extremal Graph Theory*. Academic Press, 1978.