Algorithms and Uncertainty, Summer 2021 Lecture 11 (5 pages)

Online Steiner Tree in Probabilistic Models

Thomas Kesselheim Last Update: May 12, 2021

Today, we will come back to the Online Steiner Tree problem. We have already seen what results one can obtain in the all-worst-case competitive analysis. But what if we are less pessimistic? We will consider the two approaches we already used for optimal stopping: First, we will ask what we can do when we know in advance the probability distributions the input is drawn from. Then, we will instead consider the setting in which we know nothing in advance but the input is revealed in random order.

1 Recap: Online Steiner Tree

Recall the Online Steiner Tree problem. We are given a connected graph $G = (V, E)$, edge weights $w_e \geq 0$ for $e \in E$, a root $r \in V$. A sequence of terminals $t_1, \ldots, t_k \in V$ is revealed to us one at a time. We maintain a set $S \subseteq E$ of edges. Whenever a terminal gets revealed, we have to connect it to the previous terminals or the root. That is, when we see t_i for the first time, we immediately and irrevocably add edges to S so as to connect t_i to one of $r, \ldots, t_{i-1}$.

Without loss of generality, we can assume that $G = (V, E)$ is a complete graph. We can also assume that the weights w_e fulfill the triangle inequality. That is, $w_{\{u,v\}} \leq w_{\{u,x\}} + w_{\{x,v\}}$ for all $u, v, x \in V$. This is without loss of generality because instead of taking an edge $\{u, v\}$ we could take all edges on a shortest path between u and v.

We will again use that Steiner trees can be approximated by minimum spanning trees. Such a spanning tree only uses edges between the nodes in the set $\{r\} \cup T$ and no edges to other vertices (called Steiner vertices). Let $\mathrm{MST}(T) \subseteq E$ be the minimum spanning tree on $G|_{\{r\}\cup T}$ and let $\mathrm{Steiner}(T) \subseteq E$ be the optimal Steiner tree connecting $\{r\} \cup T$. Also today we will again make use of the following lemma.

Lemma 11.1. *A minimum spanning tree on $G|_{\{r\}\cup T}$ is a 2-approximation for the min-cost Steiner tree on $\{r\} \cup T$, formally*

$$w(\mathrm{MST}(T)) \leq 2 \cdot w(\mathrm{Steiner}(T))$$

We showed that the greedy algorithm is strictly $O(\log k)$-competitive. That is, on any sequence σ of (at most) k terminals, we have

$$c(\mathrm{ALG}(\sigma)) \leq O(\log k) \cdot c(\mathrm{OPT}(\sigma)) \ ,$$

where $c(\mathrm{ALG}(\sigma))$ is the cost that the algorithm incurs and $c(\mathrm{OPT}(\sigma))$ is the optimal offline solution.

Today, our benchmark will also be the optimal offline solution, although it might be random.

2 Known Distributions

In our first probabilistic model, we assume that $t_1, \ldots, t_k$ are random variables. They are drawn independently from probability distribution, which are known by the algorithm in advance. That is, for step i, we have a vector of probabilities $(p_v^{(i)})_{v \in V}$, where $p_v^{(t)}$ denotes the probability that v arrives as a terminal in any step. We also assume that k is known.

In principle, you could model this setting as a Markov decision process but the state space would again be enormous. Also our question is different today: We are not interested in the optimal policy but we would like to see how close we can get to the optimal offline solution.

To this end, we consider the following GreedyWithSample algorithm. Draw $t'_1, \dots t'_k$ from the probability distribution. Let $T' = \{t'_1, \dots t'_k\}$. Initialize $S = \text{MST}(T')$. Now, like in the Greedy Algorithm, connect arriving new terminals to the current Steiner tree via a shortest path. That is, when t_i arrives, let v_i be the one of the vertices $r, t'_1, \dots, t'_k, t_1, \dots, t_{i-1}$ that is closest to t_i. (Note that this can also be t_i itself because we draw from our distribution with replacement.) Add $e_i = \{t_i, v_i\}$ to S.

For this algorithm, we get a similar guarantee as in competitive analysis but now taken in expectation over all possible inputs from the distribution and not point-wise for every possible sequence. The big difference is that we only lose a constant factor compared to the expected cost of the optimal offline solution.

Theorem 11.2. GreedyWithSample *guarantees* $\mathbf{E}\left[c(ALG(\sigma))\right] \leq 4 \cdot \mathbf{E}\left[c(OPT(\sigma))\right]$*, where the expectation is taken over* σ*.*

Proof. Our algorithm's cost consists of two components: The cost of $\text{MST}(T')$ and then the cost of $e_1, \dots, e_k$.

Observe that T and T' are identically distributed. Therefore

$$\mathbf{E}\left[w(\text{MST}(T'))\right] = \mathbf{E}\left[w(\text{MST}(T))\right] \leq 2\mathbf{E}\left[w(\text{Steiner}(T)\right] \ .$$

In the remainder of the proof, we will show that

$$\mathbf{E}\left[\sum_{i=1}^{k} w(e_i)\right] \leq \mathbf{E}\left[w(\text{MST}(T'))\right] \ .$$

Together this then implies

$$\mathbf{E}\left[c(\text{ALG}(\sigma))\right] = \mathbf{E}\left[w(\text{MST}(T')) + \sum_{i=1}^{k} w(e_i)\right] \leq 4\mathbf{E}\left[w(\text{Steiner}(T)\right] = 4\mathbf{E}\left[c(\text{OPT}(\sigma))\right] \ .$$

In order to upper-bound $w(e_i)$, observe the following. Instead of connecting t_i to the closest of $r, t'_1, \dots, t'_k, t_1, \dots, t_{i-1}$, we might also confine ourselves to $r, t'_1, \dots, t'_{i-1}, t'_{i+1}, \dots, t'_k$, meaning that we connect to the closest of these k vertices. This can only increase the distance.

More formally, let us write $d(T'_{-i}, t_i) = \min_{v \in T'_{-i}} w(\{v, t_i\})$, where $T'_{-i} = \{t'_j \mid j \neq i\} \cup \{r\}$. Now we can write $\mathbf{E}\left[w(e_i)\right] \leq \mathbf{E}\left[d(T'_{-i}, t_i)\right] = \mathbf{E}\left[d(T'_{-i}, t'_i)\right]$. The last step is crucial: t_i and t'_i are identically distributed; all other t'_j are independent. So, we can swap the two. Now the following lemma comes to our rescue.

Lemma 11.3. *Let* $u_1, \dots, u_\ell \in V$ *be any* ℓ *not necessarily distinct vertices. Let* $U_{-i} = \{u_j \mid j \neq i\} \cup \{r\}$*. Then for* $d(U_i, u_i) = \min_{v \in U_i} w(\{v, u_i\})$*, we have* $\sum_{i=1}^{\ell} d(U_{-i}, u_i) \leq w(\text{MST}(\{u_1, \dots, u_\ell\}))$*.*

Proof. For each u_i, we will define $v_i \in U_i$, which helps us bound $d(U_{-i}, u_i) \leq d(u_i, v_i)$. To this end, consider $\text{MST}(\{u_1, \dots, u_\ell\})$ as a tree rooted at r. If $u_i \in U_{-i}$ (this can happen because $u_1, \dots, u_\ell$ are not necessarily distinct), define $v_i = u_i$. Otherwise, if $u_i \notin U_{-i}$, define $v_i \in U_{-i}$ as the parent of u_i in $\text{MST}(\{u_1, \dots, u_\ell\})$.

Note that each edge from $\text{MST}(\{u_1, \dots, u_\ell\})$ occurs at most once as (u_i, v_i).

Therefore

$$\sum_{i=1}^{\ell} d(U_{-i}, u_i) \leq \sum_{i=1}^{\ell} d(u_i, v_i) \leq w(\mathrm{MST}(\{u_1, \ldots, u_\ell\})) \ .$$

□

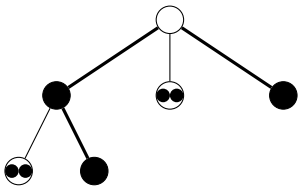


Figure 1: Illustration of Lemma 11.3 with $\ell = 7$. The three thick edges appear as (u_i, v_i), connecting the three vertices, which appear only once. The others appear twice, so they are connected to themselves.

Combining the above observations with Lemma 11.3, we now get

$$\mathbf{E}\left[\sum_{i=1}^{k} w(e_i)\right] = \sum_{i=1}^{k} \mathbf{E}\left[w(e_i)\right] \leq \sum_{i=1}^{k} \mathbf{E}\left[d(T'_{-i}, t_i)\right] = \mathbf{E}\left[\sum_{i=1}^{k} d(T'_{-i}, t_i)\right] \leq \mathbf{E}\left[w(\mathrm{MST}(\{t'_1, \ldots, t'_k\}))\right] \ .$$

Overall, this proves the theorem. □

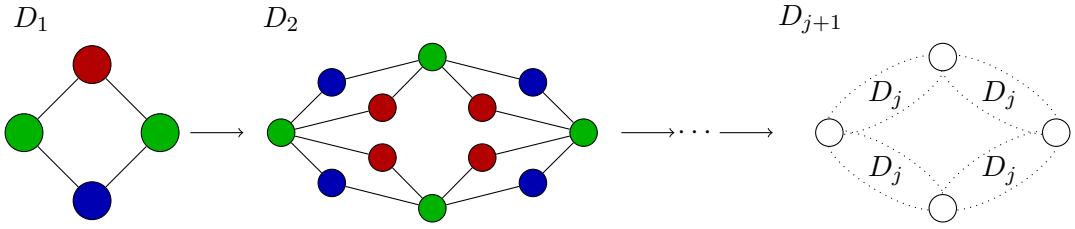
3 Random Order

Now, we will turn to a different setting. Namely, there is a fixed sequence of terminals $t_1, \ldots, t_k$, which will be presented to us but in a random order. One could hope that this would allow us to obtain improved bounds compared to a worst-case order like in the optimal-stopping problem. However, this is not true here. Even with random order, we cannot beat the $O(\log k)$-factor.

Theorem 11.4. *There is no algorithm, which guarantees* $\mathbf{E}\left[c(ALG(\sigma))\right] \leq \alpha c(OPT(\sigma))$ *for* $\alpha = o(\log k)$*, even when the sequence is presented in random order and when the algorithm is allowed to use randomization.*

Proof. Without loss of generality, we assume the algorithm whenever it picks edges it only chooses a simple path which does not contain any terminals except its endpoint. This is without loss because removing any further edges from the algorithm's choice will only decrease its cost.

We will construct our instances again based on diamond graphs. Recall that we define the *diamond graph* D_j of level j recursively: D_{j+1} is defined by connecting 4 copies of D_j as depicted in the following picture.



Note that each node is the central node of the top or bottom path in some D_j. Call this j the *level* of this vertex. So D_j contains 2 vertices of level j, 8 vertices of level $j-1$, and, more generally, $2 \cdot 4^i$ vertices of level $j-i$.

Our goal is to define a graph and a sequence of terminals of length at most k in this graph. We only consider the case that k is a power of 16 and we let $\ell = \frac{1}{2}\log_4 k$ (which is an integer then) and consider the graph D_ℓ.

To define the sequence of terminals, we draw one path from the left to the right node in D_ℓ uniformly at random. Note that this path crosses through 2^i copies of a diamond graph of level $D_{\ell-i}$ and therefore also through 2^i nodes of level $\ell - i$ for $i \in \{0, \dots, \ell-1\}$.

In our sequence, we will not simply request each node on this path once but multiple times. Namely, we request each of the nodes of level $\ell - i$ in the path $4^{\ell-i}$ times. This has no consequence for the offline optimum because repeated terminals still have to be connected only once. Observe that the length of our sequence is now $\sum_{i=0}^{\ell} 2^i \cdot 4^{\ell-i} = 4^\ell \sum_{i=0}^{\ell} \left(\frac{2}{4}\right)^i \le 2 \cdot 4^\ell = k$.

These repetitions have a crucial effect on the random order, which make it appear close enough to a worst-case order. The idea is as follows: When the first copy of a level-j vertex on the path arrives, it is likely that none of the vertices of levels $1, \dots, j-1$ in the respective level-$(j+1)$ diamond subgraph have arrived so far. In such a case, the algorithm has had no information regarding the terminals in this subgraph before. In particular, it does not know whether the level-j vertex will be the one at the top or at the bottom. Therefore, it is likely that it has to pay the cost of connecting this vertex to one of the endpoints of the subgraph.

To make this argument formal, let $v_{j,b}$ be the b-th level-j vertex on the path (viewed from the left, not as it appears in the sequence). Note that this is a random variable as we are requesting a random path. Let $X_{j,b}$ be the cost that the algorithm incurs when connecting this vertex. This random variable depends not only on the path but also on the random order, which shuffles the input. Furthermore, we can allow the algorithm to be randomized; this does not change this argument.

We now claim that $\mathbf{E}[X_{j,b}] \ge 2^{j-3}$ for all j and b.

This will cause that the overall cost of the algorithm is at least

$$\mathbf{E}\left[\sum_{j=1}^{\ell}\sum_{b=1}^{2^{\ell-j}} X_{j,b}\right] = \sum_{j=1}^{\ell}\sum_{b=1}^{2^{\ell-j}} \mathbf{E}[X_{j,b}] \ge \sum_{j=1}^{\ell} 2^{\ell-j}2^{j-3} = \frac{\ell}{8}2^\ell \ .$$

Note that the offline optimum will always have cost 2^ℓ because it is simply the one path that we started from.

To bound $\mathbf{E}[X_{j,b}]$, we observe that $v_{j,b}$ will be a middle node of a diamond graph of level $j+1$. The path connects the left and the right vertex of this graph. More precisely, there are 2^i terminals of level $j-i$ between the left and right vertex for $i \in \{0, \dots, j-1\}$.

Therefore, with probability

$$\frac{4^j}{\sum_{i=0}^{j-1} 2^i 4^{j-i}} \ge \frac{1}{\sum_{i=0}^{\infty}\left(\frac{1}{2}\right)^i} = \frac{1}{2}$$

one copy of $v_{j,b}$ is the first terminal to arrive inside this D_{j+1}.

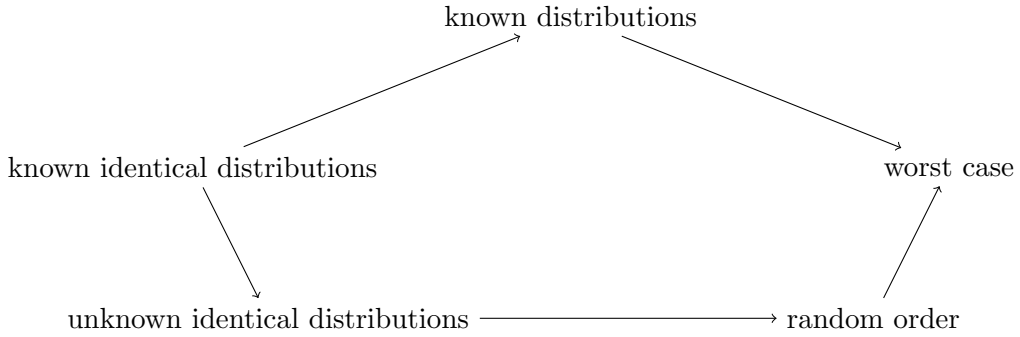
The key observation is as follows. Before any of the other terminals in this D_{j+1} arrive, we have no information whether $v_{j,b}$ is the middle vertex at the top or at the bottom. Both is equally likely. Because there were no terminals requested inside this D_{j+1} so far, all chosen edges (if any) will either be in the top or in the bottom half.

So, with probability at least $\frac{1}{2}\cdot\frac{1}{2}$, the algorithm will have to pay at least the cost of connecting $v_{j,b}$ to one of the level-$(j+1)$ vertices. The cost of this is 2^{j-1}.

So, overall $\mathbf{E}[X_{j,b}] \ge \frac{1}{2}\cdot\frac{1}{2}\cdot 2^{j-1} = 2^{j-3}$. □

4 Overview: Probabilistic Online Models

We have seen two different online models, which introduce some randomness. There are a couple of others as depicted in the following diagram.



Here an arrow $a \longrightarrow b$ indicates that a is a special case of b or in other words that any algorithm solving b can also solve a with the same guarantee — some of these are trivial, others nice exercises. One should also mention that case of *unknown, non-identical distributions* is meaningless because there is nothing that could be observed and learned. So, it is as difficult as the worst case.

While these relations hold generally, one can also show in the case of Online Steiner Tree specifically that even with unknown identical distributions the same $\Omega(\log k)$-lower bound still applies. So the frontier seems to be between cases in which we know the distributions beforehand and in which we do not know them.

Reference

- Naveen Garg, Anupam Gupta, Stefano Leonardi, Piotr Sankowski: Stochastic analyses for online combinatorial optimization problems. SODA 2008.