Learning-Augmented Algorithms for Online Steiner Tree

Chenyang Xu [1,*] **and Benjamin Moseley** [2]

[1] College of Computer Science, Zhejiang University
[2] Tepper School of Business, Carnegie Mellon University
xcy1995@zju.edu.cn, moseleyb@andrew.cmu.edu

Abstract

This paper considers the recently popular beyond-worst-case algorithm analysis model which integrates machine-learned predictions with online algorithm design. We consider the online Steiner tree problem in this model for both directed and undirected graphs. Steiner tree is known to have strong lower bounds in the online setting and any algorithm's worst-case guarantee is far from desirable.

This paper considers algorithms that predict which terminal arrives online. The predictions may be incorrect and the algorithms' performance is parameterized by the number of incorrectly predicted terminals. These guarantees ensure that algorithms break through the online lower bounds with good predictions and the competitive ratio gracefully degrades as the prediction error grows. We then observe that the theory is predictive of what will occur empirically. We show on graphs where terminals are drawn from a distribution, the new online algorithms have strong performance even with modestly correct predictions.

Introduction

An emerging line of work on beyond-worst-case algorithms makes use of machine learning for algorithmic design. This line of work suggests that there is an opportunity to advance the area of beyond-worst-case algorithmics and analysis by augmenting combinatorial algorithms with machine learned predictions. Such algorithms perform better than worst-case bounds with accurate predictions while retaining the worst-case guarantees even with erroneous predictions. There has been significant interest in this area (e.g. (Gupta and Roughgarden 2017; Balcan et al. 2018; Balcan, Dick, and White 2018; Chawla et al. 2020; Kraska et al. 2018; Lykouris and Vassilvtiskii 2018; Purohit, Svitkina, and Kumar 2018; Lattanzi et al. 2020)).

Online Learning-Augmented Algorithms. This paper considers the augmenting model in the online setting where algorithms make decisions over time without knowledge of the future. In this model, an algorithm is given access to a learned prediction about the problem instance. The learned prediction is error prone and the performance of the algorithm is expected to be bounded in terms of the prediction's quality. The quality measure is prediction specific. The performance measure is the *competitive ratio* where an algorithm is c-competitive if the algorithm's objective value is at most a c factor larger than the optimal objective value *for every input*. In the learning-augmented algorithms model, finding appropriate parameters to predict and making the algorithm robust to the prediction error are usually key algorithmic challenges.

Many online problems have been considered in this context, such as caching (Lykouris and Vassilvtiskii 2018; Rohatgi 2020; Jiang, Panigrahi, and Sun 2020; Wei 2020), page migration (Indyk et al. 2020), metrical task systems (Antoniadis et al. 2020a), ski rental (Purohit, Svitkina, and Kumar 2018; Gollapudi and Panigrahi 2019; Anand, Ge, and Panigrahi 2020), scheduling (Purohit, Svitkina, and Kumar 2018; Im et al. 2021), load balancing (Lattanzi et al. 2020), online linear optimization (Bhaskara et al. 2020), online flow allocation (Lavastida et al. 2021), speed scaling (Bamas et al. 2020), set cover (Bamas, Maggiori, and Svensson 2020), and bipartite matching and secretary problems (Antoniadis et al. 2020b).

The Steiner Tree Problem. Steiner tree is one of the most fundamental combinatorial optimization problems. For undirected Steiner tree, there is an undirected graph $G = (V, E)$ where each edge $e \in E$ has a cost c_e and a terminal set $T \subseteq V$. We need to buy edges in E such that all terminals are connected via the bought edges and the goal is to minimize the total cost of the bought edges. For the directed case, the edges are directed and there is a root node r. In this problem all of the terminals must have a directed path to the root via the edges bought.

Theoretically, the problem has been of interest to the community for decades, starting with the inclusion in Karp's 21 NP-Complete problems (Karp 1972). Since then, it has been studied extensively in approximation algorithm design (Kou, Markowsky, and Berman 1981; Takakashi 1980; Wu, Widmayer, and Wong 1986; Byrka et al. 2010), stochastic algorithms (Gupta and Pál 2005; Gupta, Hajiaghayi, and Kumar 2007; Kurz, Mutzel, and Zey 2012; Leitner et al. 2018) and online algorithms (Imase and Waxman 1991; Berman and Coulston 1997; Angelopoulos 2008, 2009). Practically, the Steiner tree problem is fundamental for many network problems such as fiber optic networks (Bachhiesl et al. 2002),

*The corresponding author.

social networks (Chiang et al. 2013; Lappas et al. 2010), and biological networks (Sadeghi and Fröhlich 2013). This problem has so many uses practically, that recently there have been competitions to find fast algorithms for it and its variants, including the 11th DIMACS Implementation Challenge (2014) and the 3rd Parameterized Algorithms and Computational Experiments (PACE) Challenge (2018).

This paper focuses on the online version of Steiner tree. In this case, the graph G is known in advance, meaning that the edges that can be bought are completely known as well as all the nodes in the graph. However, the nodes that actually are the terminals T are unknown. The terminals in T arrive one at a time. Let $t_1, t_2, \dots t_k$ be the arriving order of terminals, where $k = |T|$. When terminal t_i arrives, it must immediately be connected to $t_1, t_2, \dots t_{i-1}$ by buying edges of G and once an edge is bought, it is irrevocable. The goal is to minimize the total cost.

The online problem occurs often in practice. For instance, when building a network often new nodes are added to the network over time. Not knowing which terminals will arrive makes the problem inherently hard. The algorithm with the best worst-case guarantees is the simple greedy algorithm (Imase and Waxman 1991), which always chooses to connect an arriving node via the cheapest feasible path. The competitive ratios of the greedy algorithm on undirected graphs and directed graphs are, respectively, $\Theta(\log k)$ and $\Theta(k)$, which are the best possible using worst-case analysis (see (Imase and Waxman 1991; Westbrook and Yan 1995)). However, these results are far from desirable. The question thus looms, is there the potential to go beyond worst-case lower bounds in the learning-augmented algorithms for online Steiner tree?

Results

We consider the online Steiner tree problem in the learning-augmented model. The prediction is defined to be the set of terminals. That is, the algorithm is supplied with a set of terminals $\hat{T}$ at the beginning of time. Some of these may be incorrect. Define the prediction error η to be the number of incorrectly predicted terminals. Then the actual terminals in T arrive online. This paper shows the following results, breaking through worst-case lower bounds.

- In the undirected case, we propose an $O(\log \eta)$-competitive algorithm. That is, with accurate predictions, the algorithm is *constant competitive*. Then with the worst predictions, the competitive ratio is $O(\log k)$, *matching the best worst-case bound*. Between, the algorithm has slow degradation of performance in terms of the prediction error. We further show that any algorithm has competitive ratio $\Omega(\log \eta)$ with this prediction and thus our algorithm is the best possible online algorithm using this prediction.
- In the directed case, we give an algorithm that is $O(\eta + \log k)$-competitive. With near perfect predictions, the algorithm is $O(\log k)$-competitive, which is exponentially better than the worst-case lower bound $\Omega(k)$. With a large prediction error, the algorithm matches the $O(k)$ bound of the best worst-case algorithm. Between, the algorithm has slow degradation of performance in terms of the error as in the undirected case. As in the undirected case, we show that any algorithm has competitive ratio $\Omega(\eta)$ with this prediction. Our algorithm is close to the best possible when using this prediction.

The next question is if these theoretical results predict what will occur empirically on real graphs. For the undirected case we show that with modestly accurate predictions, the algorithms indeed can outperform the baseline. Then the performance degrades as there is more error in the prediction, never becoming much worse than the baseline. These empirical results corroborate the theory. Moreover, we give a learning algorithm that is able to learn predictions from a small number of sample instances such that our Steiner tree algorithms have strong performance.

Online Undirected Steiner Tree

For the brevity of the algorithms' statement and analysis, we make two assumptions. First, we assume that G is a complete graph in metric space. This can be assumed by taking the metric completion of any input graph and is standard for the Steiner tree problem. Second, the predicted terminal set $\hat{T}$ and the real terminal set T share the same size k. The discussion about this assumption is provided in the full version of this paper. We aim to show the following theorem in this section.

Theorem 1. *Given a predicted terminal set $\hat{T}$, there exists an algorithm with competitive ratio at most $O(\log \eta)$, where $\eta := k - |T \cap \hat{T}|$.*

Preliminaries

The input is an undirected graph $G = (V, E)$, where each edge e has cost $c_e \geq 0$, and a terminal set $T \subseteq V$ that arrives online. Recall $k := |T| = |\hat{T}|$. When a terminal t arrives, we must buy some edges such that it is connected with all previous terminals in the subgraph formed by bought edges. The goal is to minimize the total cost of the bought edges.

In the analysis, we will leverage results on the online greedy algorithm. The following theorem was shown in (Imase and Waxman 1991). The traditional online greedy algorithm maintains a tree T connecting all the terminals. This tree is initialized to $\emptyset$. Then when a terminal t arrives, the edges on the shortest path from t to any node in T will be added into T.

Theorem 2 ((Imase and Waxman 1991))**.** *The online greedy algorithm is $O(\log k)$-competitive.*

We will also use the following properties of minimum spanning trees.

Lemma 3. *Consider an offline Steiner tree instance. A minimum spanning tree $\mathrm{MST}(T)$ on terminals is a 2-approximated solution (Kou, Markowsky, and Berman 1981). In addition, for any edge e, if $\{e\} \cup \mathrm{MST}(T)$ contains a cycle, c_e is the maximum edge cost in the cycle (Schrijver 2003).*

Warm-up: Analysis of a Simple Online Algorithm

Towards proving Theorem 1, we first introduce a simple and natural algorithm whose competitive ratio is $O(\eta)$. This is a far worse guarantee than the algorithm we develop, but it will help build our techniques and give the intuition.

Intuitively, if the prediction is error-free, the instance becomes an offline problem. Several constant approximation algorithms can be employed for the offline case. For example, we compute a minimum spanning tree $\mathrm{MST}(\hat{T})$ on the accurate predicted terminal set $\hat{T}$ and each time when a new terminal arrives, connect it with all previous terminals only using the edges in $\mathrm{MST}(\hat{T})$. This algorithm obtains a competitive ratio 2 if $\hat{T} = T$.

Inspired by this, a natural online algorithm is the following. This algorithm has poor performance when the error in the predictions is large. This will then lead us to develop a more robust algorithm.

Online Algorithm with Predicted Terminals (OAPT): Let $\hat{T}$ be the predicted set of terminals and $\mathrm{MST}(\hat{T})$ be the minimum spanning tree on $\hat{T}$. Let T_i be the first set of i terminals that arrive online. T_k contains all online terminals.

Initialize $A = \emptyset$ to be the tree that the algorithm will construct connecting the online terminals. The algorithm returns the set of edges in A after all terminals arrive. We divide the edges of $A = A_1 \cup A_2$ into two sets, A_1 and A_2 depending on the case that causes us to add edges to A. Consider when terminal t_i arrives.

- **Case 1:** If $t_i \notin \hat{T}$ or t_i is the first terminal in $\hat{T}$ to arrive, add to A_1 the shortest edge in G connecting t_i to terminals T_{i-1} that have arrived. No edge is bought if this is the first terminal that arrives.
- **Case 2:** Otherwise, add the shortest path in $\mathrm{MST}(\hat{T})$ to A_2 which connects t_i to a terminal in $\hat{T} \cap T_{i-1}$. In other words, buy the shortest path in $\mathrm{MST}(\hat{T})$ connecting t_i to a *predicted* terminal that has previously arrived.

Our goal is to show that the competitive ratio of this algorithm is exactly $\Theta(\eta)$.

Theorem 4. *The competitive ratio of OAPT is $\Theta(\eta)$.*

First we observe that the algorithm is no better than $\Omega(\eta)$-competitive. This lower bound will motivate the design of a more robust algorithm in the next section.

Lemma 5. *The competitive ratio of OAPT is $\Omega(\eta)$.*

To prove Lemma 5, we first construct an instance and then show that algorithm OAPT is η-competitive on it. The instance is shown in Fig. 1. Due to space, the detailed proof is omitted in this version.

Next we prove the upper bound of the algorithm's performance. The solution A is partitioned into two sets A_1 and A_2. We bound the cost of these sets separately. The following lemma bounds the cost of A_1. Essentially, these edges do not cost most than $O(\log \eta)$ because there are at most $\eta+1$ terminals that contribute to edges in A_1 and their cost is bounded by running the traditional online greedy algorithm on these terminals, which is logarithmically competitive.

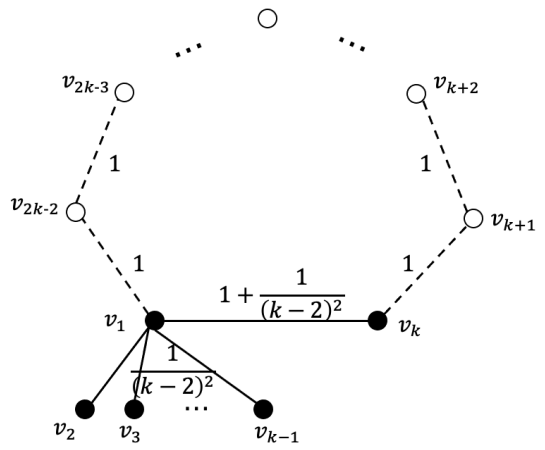


Figure 1: An online Steiner tree instance. The graph has $2k-2$ nodes and $2k-2$ edges. The cost of edge (v_1, v_k) is $1 + 1/(k-2)^2$ while each edge between v_1 and v_i ($2 \le i \le k-1$) has a cost $1/(k-2)^2$. The cost of each dash edge is 1. The terminal set T consists of $v_1, v_2, ..., v_k$ while the prediction is $\hat{T} = \{v_1, v_k, v_{k+1}, ..., v_{2k-2}\}$.

Lemma 6. *Let $c(A_1)$ be the total cost of edges in A_1. We have $c(A_1) \le O(\log \eta)\mathrm{OPT}$, where* OPT *is the optimal objective value.*

Proof. Let T' be the terminals which get connected to the tree via Case (1). By definition of η, we have $|T'| \le \eta + 1$. This is because all terminals that are in $T \setminus \hat{T}$ are in Case (1) and these contribute to η. The only other terminal that executes Case (1) is the first terminal in $\hat{T}$ that arrives.

Consider a new problem instance $\mathcal{I}'$ where the terminal set is T'. In this new instance, the graph along with edge costs remains the same. Let $\mathrm{OPT}(\mathcal{I}')$ be the optimal cost of this new instance and OPT the optimal cost of the original problem. Notice that $\mathrm{OPT}(\mathcal{I}') \le \mathrm{OPT}$ as $T' \subseteq T$ and OPT represents a feasible solution for the new instance.

We know that the cost of the traditional online greedy algorithm is at most $O(\log(|T'|))\mathrm{OPT}(\mathcal{I}')$ because it is $\log(|T'|)$ competitive on the instance $\mathrm{OPT}(\mathcal{I}')$ by Theorem 2. This holds no matter the arriving order of the terminals. Let $c(\texttt{Greedy})$ be the cost of the greedy algorithm if the terminals in T' arrive consistent with the arriving order for the original problem. We have that $c(\texttt{Greedy}) \le O(\log(|T'|))\mathrm{OPT}(\mathcal{I}') \le O(\log(\eta))\mathrm{OPT}(\mathcal{I}') \le O(\log(\eta))\mathrm{OPT}$.

Notice that the shortest edge from t_i to T_{i-1} is definitely at most the shortest edge from t_i to $T_{i-1} \cap T'$. Thus, $c(A_1) \le c(\texttt{Greedy}) \le O(\log(\eta))\mathrm{OPT}$. □

Now we focus on the second set A_2. These terminals potentially cause the bulk of the cost for the algorithm. We will bound $c(A_2)$ by $O(\eta)\mathrm{OPT}$, which proves Theorem 4. We first show that for each edge $e \in A_2$ has cost at most OPT. Notice that this is not enough to prove $c(A_2) \le O(\eta)\mathrm{OPT}$ because the number of edges in A_2 could be as large as $k-1$. For the remainder of this section, let P_i denote the path added into A_2 in iteration i.

Lemma 7. *Each edge in A_2 has cost at most* OPT.

Proof. Consider when terminal t_i arrives, the algorithm executes Case (2) and the path $P_i \neq \emptyset$. Notice that if Case (2) is executed then there is a terminal in $t_j \in \hat{T} \cap T_{i-1}$ that has arrived before t_i. Moreover, for any terminal $t_j \in \hat{T} \cap T_{i-1}$, the cost of edge (t_i, t_j) is at most OPT because these two nodes are connected in the optimal solution and $c(t_i, t_j)$ is the minimum cost to connect them. To show the lemma, we show that for any edge $e \in P_i$, $c_e \leq c(t_i, t_j)$. This then bounds the cost of any edge in A_2 by OPT.

Fix the terminal $t_j \in \hat{T} \cap T_{i-1}$ that t_i connects to using path P_i. If the edge (t_i, t_j) is in $\mathrm{MST}(\hat{T})$ then this will be the unique edge in P_i. If (t_i, t_j) is not in $\mathrm{MST}(\hat{T})$ then by Lemma 3 every edge on the cycle $P_i \cup \{(t_i, t_j)\}$ has cost at most $c(t_i, t_j) \leq$ OPT. □

We are ready to bound the cost of the edges in A_2.

Lemma 8. *The edges of A_2 can be partitioned into two sets B_1 and B_2, where $c(B_1) \leq$ OPT and $|B_2| \leq \eta$. Moreover, the total cost of edges in A_2 is at most $O(\eta)$*OPT.

Proof. We begin by partitioning the edges of A_2 into two sets B_1 and B_2. Let E' contain the edges in $\mathrm{MST}(\hat{T}) \cap \mathrm{MST}(\hat{T} \cap T)$. Initialize $S = \mathrm{MST}(\hat{T})$. The set S will always be a spanning tree of $\hat{T}$. We do the following iteratively. For each edge $e \in \mathrm{MST}(\hat{T} \cap T) \setminus \mathrm{MST}(\hat{T})$, we add it to S and remove an arbitrary edge $e' \in \mathrm{MST}(\hat{T}) \setminus \mathrm{MST}(\hat{T} \cap T)$ from S that forms a cycle. The removed edge e' is added to E'. Set $B_1 = E' \cap A_2$ and $B_2 = A_2 \setminus E'$.

Intuitively, the above procedure obtains a spanning tree S of $\hat{T}$ by replacing some edges in $\mathrm{MST}(\hat{T})$ that are not in $\mathrm{MST}(\hat{T}) \cap \mathrm{MST}(\hat{T} \cap T)$ with the edges in $\mathrm{MST}(\hat{T} \cap T)$. We have that $c(E') \leq c(\mathrm{MST}(\hat{T} \cap T))$. This is because $c(e) \geq c(e')$ in each step of the algorithm by definition of $\mathrm{MST}(\hat{T})$ and Lemma 3. Knowing $c(\mathrm{MST}(\hat{T} \cap T)) \leq$ OPT, we see that $c(B_1) \leq c(E') \leq$ OPT.

According to the algorithm, the number of edges in E' is exactly the same as the number of edges in $\mathrm{MST}(\hat{T} \cap T)$. In other words, $|E'| = k - \eta - 1$ and $|\mathrm{MST}(\hat{T}) \setminus E'| = \eta$. Since A_2 is a subset of $\mathrm{MST}(\hat{T})$, $|B_2| \leq |\mathrm{MST}(\hat{T}) \setminus E'| = \eta$. Namely, the number of edges in the second partition is at most η. Using Lemma 7, we have $c(B_2) \leq \eta$OPT, completing the proof of this lemma.

□

Proof of Theorem 4. The theorem can be proved directly by Lemma 6 and Lemma 8: $c(A) \leq c(A_1) + c(A_2) \leq O(\log \eta)$OPT $+ O(\eta)$OPT $= O(\eta)$OPT. The lower bound in the theorem is given in Lemma 5. Altogether, we have the main theorem. □

An Improved Online Algorithm Leveraging Predictions

In this section, we will build on the simple algorithm to give a more robust online algorithm that has a competitive ratio of $O(\log \eta)$. Notice that in the prior proof, the large cost arises due to the edges that are added in Case (2), especially the edges in $B_2 = A_2 \setminus E'$ in proof of the final lemma. The new algorithm is designed to mitigate this cost.

Improved Online Algorithm with Predicted Terminals (IOAPT): Let $\hat{T}$ be the predicted set of terminals and $\mathrm{MST}(\hat{T})$ be the minimum spanning tree on $\hat{T}$. Let T_i be the first set of i terminals that arrive online. T_k contains all online terminals.

Initialize $A = \emptyset$ to be the subgraph that the algorithm will construct connecting the online terminals. The algorithm returns the set of edges in A after all terminals arrive. We divide the edges of $A = A_1 \cup A_2$ into two sets, A_1 and A_2 depending on the case that causes us to add edges to A. Consider when terminal t_i arrives.

- **Case 1:** If $t_i \notin \hat{T}$ or t_i is the first terminal in $\hat{T}$ to arrive, add to A_1 the shortest edge in G connecting t_i to terminals T_{i-1} that have arrived. No edge is bought if this is the first terminal that arrives.
- **Case 2:** Otherwise, find the shortest path P_i connecting t_i to $\hat{T} \cap T_{i-1}$ in $\mathrm{MST}(\hat{T})$. Use e_i to denote the shortest edge connecting t_i to $\hat{T} \cap T_{i-1}$ in G. We add to A_2 a sub-path $P_i' \subseteq P_i$ such that its endpoints contain t_i while its total cost is in $[c_{e_i}, 2c_{e_i}]$. Next, add e_i to A_2 if t_i is not connected to the tree after adding P_i'.

Notice that in Case 2, we can always find such a sub-path P_i' due to the property of the minimum spanning tree and the assumption that G is a metric. Thus, the algorithm always computes a feasible solution. We have the following two lemmas. The proofs are identical to Lemma 7 and Lemma 8 respectively.

Lemma 9. *The cost of any edge in A_2 computed by IOAPT is at most* OPT.

Lemma 10. *The edges of A_2 can be partitioned into two sets B_1 and B_2 where $c(B_1) \leq$ OPT and $|B_2| \leq \eta$.*

With these lemmas, we can prove the theorem.

Theorem 11. *The competitive ratio of IOAPT is $O(\log \eta)$.*

Proof. The analysis of $c(A_1)$ is the same as that in OAPT. The proof of Lemma 6 immediately implies $c(A_1) \leq O(\log \eta)$OPT. Next we focus on bounding the cost of A_2.

Let $\underset{i}{\Delta}\, c(A_2)$ be the increase in $c(A_2)$ in when terminal t_i arrives. According to definition of the algorithm, we know $\underset{i}{\Delta}\, c(A_2) \leq 2c(P_i')$ and $\underset{i}{\Delta}\, c(A_2) \leq 3c_{e_i}$. Lemma 10 states the edges in $\mathrm{MST}(\hat{T})$ can be partitioned into two sets E_0 and $E_1 := \mathrm{MST}(\hat{T}) \setminus E_0$, where the cost of E_0 is at most 2OPT and the number of edges in E_1 is η.

Let T^G be the 'good' terminals that execute Case (2) and $P_i' \subseteq E_0$. Let T^B be the remaining 'bad' terminals. We see the following for the good terminals, $c(A_2) = \sum_{i \in T^G} \underset{i}{\Delta}\, c(A_2) \leq \sum_{i \in T^G} 2c(P_i') \leq 2c(E_0) \leq O(\mathrm{OPT})$. In other words, if the sub-path added in each iteration always belongs to E_0, the total cost of $c(A_2)$ is bounded by a constant factor of OPT. Say an iteration is good if the sub-path

added in it belongs to E_0. The total increment of all good iterations is at most $O(\text{OPT})$.

We use the second upper bound to analyze the cost of the bad terminals. This follows similarly to the proof of Lemma 6. Indeed, we know the following, $\sum_{i \in T^B} \Delta_i c(A_2) \leq \sum_{i \in T^B} 3c_{e_i}$. If iteration i is bad, there exists at least one edge in sub-path P'_i belonging to E_1. Since $|E_1| = \eta$, the number of bad iterations is at most η. The total cost of these iterations $\sum_{i \in T^B} 3c_{e_i}$ is at most 3 multiplied by the cost of running the greedy algorithm on the terminals in T^B. Let $\text{OPT}(T^B)$ be the optimal solution on T^B. We know that $\text{OPT} \geq \text{OPT}(T^B)$. Moreover, we know that the greedy algorithm has cost at most $O(\log(|T^B|))\text{OPT}(T^B) \leq O(\log(|T^B|))\text{OPT} \leq O(\log \eta)\text{OPT}$. Thus we have the following, $\sum_{i \in T^B} \Delta_i c(A_2) \leq O(\log \eta)\text{OPT}$. This completes the proof of Theorem 11. □

The competitive ratio $O(\log(\eta))$ approaches the worst-case bound $\log(k)$ when $\eta = k$. Here we give a stronger statement to show our algorithm optimally uses the predictions. The proof is provided in the full version of this paper.

Theorem 12. *For online undirected Steiner tree with predicted terminals, given any $\eta \geq 1$, no online algorithm has a competitive ratio better than $\Omega(\log(\eta))$.*

Improving the Performance of the Algorithm in Practice. We describe a practical modification of the algorithm. This modification ensures that the algorithm maintains its theoretical bound, while improving the performance. The observation is that the algorithm may purchase edges not needed for feasibility. Some edges added by our algorithm are purchased based on predicted terminals and they will become useless if these predicted terminals do not arrive. We can choose not to buy these edges immediately. When t_i arrives, the edges in P'_i are not bought immediately if we buy e_i. Instead, the algorithm buys the edges the first time a terminal uses them to connect to previous terminals.

Online Steiner Tree in Directed Graphs

This section considers online Steiner tree when the graph is directed. The input is a directed graph $G = (V, E)$, where each edge e has cost $c_e \geq 0$, a root vertex $r \in V$ and a terminal set $T \subseteq V$ that arrives online. This paper assumes without loss of generality, that $c_e > 1$ for any edge e. Additionally the input graph is assumed to ensure that there exists a directed path from root r to every vertex in V.

The terminals in T arrive online. When a terminal $v \in T$ arrives the algorithm must buy some edges to ensure there is a directed path from the root r to v in the subgraph induced by the bought edges. The goal is to minimize the total cost of the bought edges.

In directed graphs, the worst-case bound on the competitive ratio is $\Omega(k)$ (Westbrook and Yan 1995). Our main result shows that we can break through this bound.

Theorem 13. *Given a predicted terminal set $\hat{T}$, there exists an algorithm with competitive ratio at most $O(\log k + \eta)$, where $\eta := k - |T \cap \hat{T}|$.*

The algorithm claimed in Theorem 13 is $O(\log k)$-consistent and $O(k)$-robust, meaning that the ratio is $O(\log k)$ if $\eta = 0$ and is at most $O(k)$ for any η. The algorithm for directed graphs builds on the algorithm for undirected graphs. As before, there are two sets of edges A_1, A_2. The set A_1 contains edges that are bought because a terminal arrives that was not predicted. As in the undirected case such these edges are bought using a greedy algorithm. The edges in A_2 are bought using a different algorithm over the undirected case.

Online Algorithm with Predicted Terminals in Directed Graphs: Initialize $\lambda = 1$ to be a parameter, which is intuitively a guess of the maximum connection cost of any terminal in T. Let $\hat{T}(\lambda) := \{t \in \hat{T} \mid c(t, r) \leq \lambda\}$ be the set of predicted terminals that have a path to the root of cost at most λ. Let $\text{MDST}(\hat{T}(\lambda))$ be the minimum directed Steiner tree of $\hat{T}(\lambda)$, which can be computed by an offline optimal algorithm[1].

Initialize $A_1 = \emptyset$ and $A_2 = \emptyset$. The edges that are bought will be $A = A_1 \cup A_2$. Order the terminals such that t_i arrives before t_{i+1} and let $T_i = \{t_1, t_2, \ldots, t_i\}$ be the first i terminals to arrive. Let $\beta_i = \max_{t_j \in T_i} c(t_j, r)$ be the maximum cost of connecting a terminal in T_i directly to the root.

Consider when a terminal $t_i \in T$ arrives. If $\beta_i > \lambda$ then both increase λ by a factor 2 and update $\hat{T}(\lambda)$ and $\text{MDST}(\hat{T}(\lambda))$. Next perform one of the following.

- If $t_i \notin \hat{T}(\lambda)$ then add the shortest path from t_i to r to A_1, buying these edges.
- Otherwise, add the unique path from t_i to r in $\text{MDST}(\hat{T}(\lambda))$ to A_2.

Our goal is to show the following theorem.

Theorem 14. *The competitive ratio of the Algorithm for directed Steiner tree is $O(\log k + \eta)$.*

Before proving the theorem, we show a technical lemma.

Lemma 15. *For any λ, $c(\text{MDST}(\hat{T}(\lambda))) \leq \text{OPT} + \lambda\eta$.*

Proof. The proof idea is to construct a feasible Steiner tree of $\hat{T}(\lambda)$ whose value is at most $\text{OPT} + \lambda\eta$. Then the inequality will hold due to the optimality of $\text{MDST}(\hat{T}(\lambda))$. The feasible tree is constructed as follows: connect all terminals in $\hat{T}(\lambda) \cap T$ to the root in the same way as the optimal solution and add the shortest path from t to r for each terminal $t \in \hat{T}(\lambda) \setminus T$. The total cost of the former part is at most OPT while the latter term incurs a cost of $\lambda\eta$ since $c(t, r) \leq \lambda$ for any terminal $t \in \hat{T}(\lambda)$. Thus, the total cost of this subgraph is at most $\text{OPT} + \lambda\eta$, implying that $c(\text{MDST}(\hat{T}(\lambda))) \leq \text{OPT} + \lambda\eta$. □

[1]Noting that this problem is NP-hard and it is known to be inapproximable within a $O(\log k)$ ratio unless $P = NP$ (Dinur and Steurer 2014), we do not have efficient optimal algorithms or approximation algorithms in practice. Thus, the directed case is more for theoretical interests.

We can now prove the main theorem. Due to space, the detailed proof is omitted in this paper.

Experimental Results

This section investigates the empirical performance of the proposed algorithm OAPT and IOAPT for the undirected Steiner tree problem. The goal is to answer the following two questions:

- Robustness - How much prediction accuracy does the algorithms need to outperform the baseline algorithm empirically?
- Learnability - How many samples are required to empirically learn predictions sufficient for the algorithms to perform better than the baseline?

The baseline we compare against is the online greedy algorithm which is the best traditional online algorithm. We investigate the performance of both OAPT and IOAPT.

Setup

The experiments[2] are conducted on a machine running Ubuntu 18.04 with an i7-7800X CPU and 48 GB memory. Experiments are averaged over 10 runs. We consider two types of graphs.

Random Graphs. The number of nodes in a graph is set to be 2,000 and 50,000 edges are selected uniformly at random. The cost of each edge is an integer sampled uniformly from $[1, 1000]$. To ensure the connectivity of graphs, we add all remaining edges to form a complete graph, given the edges high cost of 100,000.

Road Graphs. The road network of Bay Area is provided by The 9th DIMACS Implementation Challenge[3] in graph format where a node denotes a point in Bay Area and the cost of an edge is the road length between the two endpoints. The original data contains 321,270 nodes and 400,086 edges. In the experiments, we employ the same sampling method as in (Moseley, Vassilvitskii, and Wang 2021) to sample connected subgraphs from this large graph. Briefly, we draw rectangles with a certain size on the road network randomly and construct a subgraph from each rectangle. The experiments employ 4 sampled subgraphs with 23512 ± 1135 nodes and 31835 ± 1815 edges. These graphs give the same trends, thus, we show one such graph in this section and others appear in the full version of the paper.

The terminal set T and the prediction $\hat{T}$ are constructed differently depending on the experiments.

Robustness to Accuracy

This experiment tests the performance of OAPT and IOAPT when the prediction accuracy varies. Recall that k is the number of terminals. We set $k = 200$ and $2,000$ respectively for random graphs and road graphs unless stated otherwise[4]. The set T of k terminals are sampled uniformly from the vertex set V. They arrive in random order.

We now construct the predictions. Let $\lambda \in [0, 1]$ be a parameter corresponding to the prediction accuracy. First, we sample a node set $\hat{T}_0$ with $k\lambda$ nodes uniformly from the terminal set T. And then another node set $\hat{T}_1$ with $k(1 - \lambda)$ nodes is sampled uniformly from the non-terminal nodes $V \setminus T$. Let $\hat{T}_0 \cup \hat{T}_1$ be the predicted terminal set $\hat{T}$. Notice that λ indicates the prediction accuracy. Thus, testing the performance of algorithms with different λ's answers the robustness question. This experiment is in Fig. 2(a) and 3(a).

Learning the Terminals

Here we construct instances where the algorithm explicitly learns the terminals. Each such instance will have a distribution over terminal sets of size k and employ random order. We will sample s training instances of k terminals $T_1, T_2, \ldots T_s$. The learning algorithm used to predict terminals is defined as follows.

The Learning Algorithm. A node v is predicted to be in $\hat{T}$ with probability $f(v)/s$ if $f(v) > \theta s$, where $f(v)$ is the number of sampled sets in which node v appears and θ is a parameter in $[0, 1]$. Note that the number of predicted terminals may not equal k.

There is a question on how to choose θ. This is done as follows. We choose an instance T_i from the training set at random and check which θ would give OAPT (IOAPT) the best performance on this instance. We then use this θ for OAPT (IOAPT) on the online instance. For efficiency, we only consider $\theta \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$.

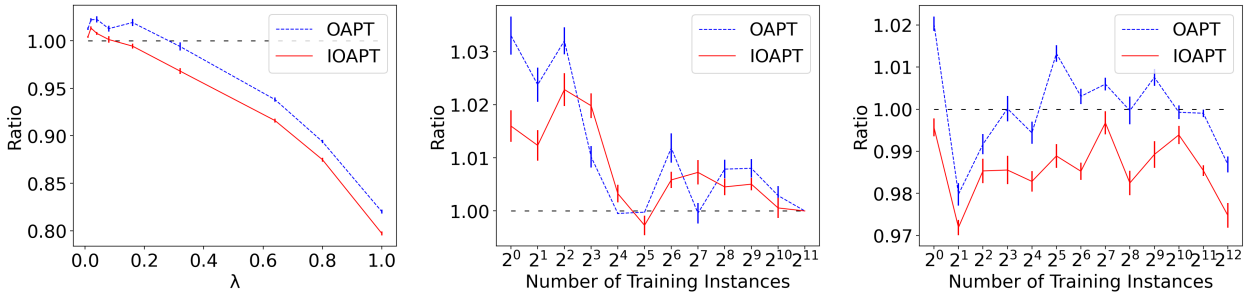
Distribution for Random Graphs. Two distributions are considered for random graphs. The first is a bad distribution where there is nothing to learn, the uniform distribution. In this case, all terminals are drawn uniformly from V. The second is called a two-class distribution where there is a set of nodes to learn. Let V_h be a small collection of nodes that will be terminals with higher probability. V_h is set to 400 nodes uniformly at random. Let $k = 200$ be the number of terminals. Half are drawn from V_h and half from $V \setminus V_h$. Here we hope the learning algorithm quickly learns V_h, and further, our algorithms can take advantage of the predictions. The results appear in Fig. 2(b) and 2(c).

Distribution for Road Graphs. This experiment is designed to model the case where terminals can appear in geographical similar locations. The graph will be clustered and a specified number of terminals will arrive per cluster following a distribution over nodes in the cluster. Use r to denote the radius of the graph. Given a parameter σ, partition all nodes into several clusters such that the radius of each cluster is at most σr. The greedy clustering algorithm (Gonzalez 1985) is used. We let $\sigma = 0.1$ in the experiments unless state otherwise. The terminal set $\hat{T}$ is obtained by picking $\lfloor 2000/x \rfloor$ random clusters and sampling x terminals uniformly from each selected cluster. We let x be 10 and 100. When $x = 10$ the distribution is harder to learn than when $x = 100$. See Fig. 3(b) and 3(c) for the results. Experiments varying parameters appear in the full version of this paper.

[2]The code is available at https://github.com/Chenyang-1995/Online-Steiner-Tree

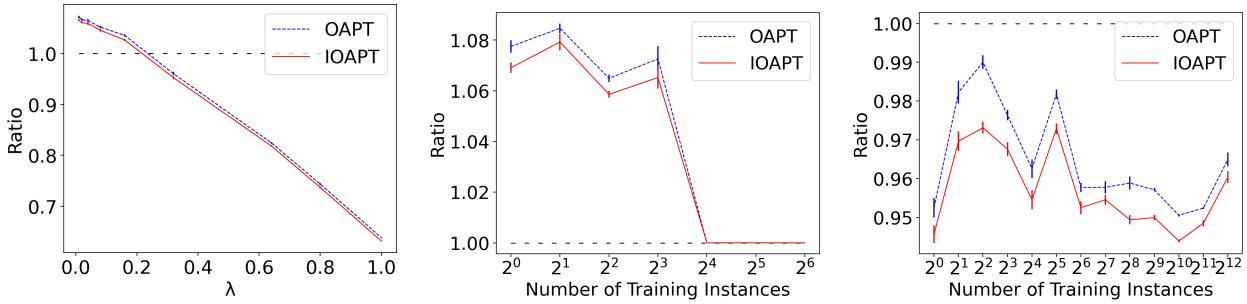
[3]http://users.diag.uniroma1.it/challenge9/download.shtml

[4]We also conduct experiments with different numbers of terminals. See this paper's full version for more results.



(a) (b) (c)

Figure 2: The experimental results on random graphs. The ratio is the algorithm's performance relative to the baseline. Fig. 2(a) shows the performance of algorithms over different λ's, corresponding to the robustness experiment. Fig. 2(b) and Fig. 2(c) are, respectively, the algorithms' performance over the number of training instances on the uniform distribution and the two-class distribution. Their corresponding prediction errors are present in this paper's full version. Note that some of the x-axes are on log-scale.



(a) (b) (c)

Figure 3: Results on road graphs. Fig. 3(a) shows the performance of algorithms over different λ's. Fig. 3(b) and Fig. 3(c) are, respectively, the algorithms' performance and prediction error over different numbers of training instances when $x = 10$ and 100. The corresponding prediction errors are present in this paper's full version.

Empirical Discussion

We see the following trends.

- Both Fig. 2(a) 3(a) show that the algorithms perform well on different graphs even with modestly correct predictions. Once about 20% of the predictions are correct, the algorithms perform better than the baseline.
- Fig. 2(b) 3(b) show the algorithms are robust for difficult distributions, which are sparse distributions where there is effectively nothing to learn. The learning algorithm will quickly realize that predictions cause negative effects and then output very few predicted terminals (see the prediction error figures in this paper's full version for more corroborating experiments). After tens of training instances, the ratios become never worse than 1.01.
- Fig. 2(c) 3(c) show the learning algorithm quickly learns good distributions. Further, both online algorithms have strong performance using the predictions. We conclude that with a small number of training samples, the learning algorithm is able to learn useful predictions sufficient for the online algorithms to outperform the baseline.

These experiments corroborate the theory. The algorithms obtain much better performance than the baseline even with modestly good predictions. If given very inaccurate predictions, the algorithms are barely worse than the baseline. Moreover, we see that only a small number of sample instances are needed for the algorithms to have competitive performance when terminals arrive from a good distribution.

Conclusion

Online Steiner tree is one of the most fundamental online network design problems. It is a special case or a subproblem of many online network design problems. Steiner tree captures the challenge of building networks online and, moreover, Steiner tree algorithms are often used as building blocks or subroutines for more general problems. As the community expands the learning augmented algorithms area into more general online network design problems, this paper provides models, and algorithmic and analysis techniques that can be leveraged for these problems.

Acknowledgements

Chenyang Xu was supported in part by Science and Technology Innovation 2030 –"The Next Generation of Artificial Intelligence" Major Project No.2018AAA0100902. Benjamin Moseley was supported in part by NSF grants CCF-1824303, CCF-1845146, CCF-2121744, CCF-1733873 and CMMI-1938909. Benjamin Moseley was additionally supported in part by a Google Research Award, an Infor Research Award, and a Carnegie Bosch Junior Faculty Chair. We thank Yuyan Wang for sharing their experimental data and thank the anonymous reviewers for their insightful comments and suggestions.

References

Anand, K.; Ge, R.; and Panigrahi, D. 2020. Customizing ML Predictions For Online Algorithms. *ICML 2020*.

Angelopoulos, S. 2008. A Near-Tight Bound for the Online Steiner Tree Problem in Graphs of Bounded Asymmetry. In *Algorithms - ESA 2008, 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*, 76–87.

Angelopoulos, S. 2009. Parameterized Analysis of Online Steiner Tree Problems. In *Adaptive, Output Sensitive, Online and Parameterized Algorithms, 19.04. - 24.04.2009*.

Antoniadis, A.; Coester, C.; Eliás, M.; Polak, A.; and Simon, B. 2020a. Online metric algorithms with untrusted predictions. *CoRR*, abs/2003.02144.

Antoniadis, A.; Gouleakis, T.; Kleer, P.; and Kolev, P. 2020b. Secretary and Online Matching Problems with Machine Learned Advice. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Bachhiesl, P.; Paulus, G.; Prossegger, M.; Werner, J.; and Stögner, H. 2002. Cost Optimized Layout of Fibre Optic Networks in the Access Net Domain. In *Operations Research Proceedings 2002, Selected Papers of the International Conference on Operations Research (SOR 2002), Klagenfurt, Austria, September 2-5, 2002*, 247–252.

Balcan, M.; Dick, T.; Sandholm, T.; and Vitercik, E. 2018. Learning to Branch. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 353–362. PMLR.

Balcan, M.; Dick, T.; and White, C. 2018. Data-Driven Clustering via Parameterized Lloyd's Families. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 10664–10674.

Bamas, É.; Maggiori, A.; Rohwedder, L.; and Svensson, O. 2020. Learning Augmented Energy Minimization via Speed Scaling. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Bamas, É.; Maggiori, A.; and Svensson, O. 2020. The Primal-Dual method for Learning Augmented Algorithms. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Berman, P.; and Coulston, C. 1997. On-line algorithms for Steiner tree problems. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, 344–353. Proceedings of the 1997 29th Annual ACM Symposium on Theory of Computing ; Conference date: 04-05-1997 Through 06-05-1997.

Bhaskara, A.; Cutkosky, A.; Kumar, R.; and Purohit, M. 2020. Online Learning with Imperfect Hints. *CoRR*, abs/2002.04726.

Byrka, J.; Grandoni, F.; Rothvoß, T.; and Sanità, L. 2010. An improved LP-based approximation for steiner tree. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, 583–592.

Chawla, S.; Gergatsouli, E.; Teng, Y.; Tzamos, C.; and Zhang, R. 2020. Pandora's Box with Correlations: Learning and Approximation. arXiv:1911.01632.

Chiang, M.; Lam, H.; Liu, Z.; and Poor, H. V. 2013. Why Steiner-tree type algorithms work for community detection. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, 187–195.

Dinur, I.; and Steurer, D. 2014. Analytical approach to parallel repetition. In Shmoys, D. B., ed., *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, 624–633. ACM.

Gollapudi, S.; and Panigrahi, D. 2019. Online Algorithms for Rent-Or-Buy with Expert Advice. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 2319–2327. PMLR.

Gonzalez, T. F. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theor. Comput. Sci.*, 38: 293–306.

Gupta, A.; Hajiaghayi, M.; and Kumar, A. 2007. Stochastic Steiner Tree with Non-uniform Inflation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX 2007, and 11th International Workshop, RANDOM 2007, Princeton, NJ, USA, August 20-22, 2007, Proceedings*, 134–148.

Gupta, A.; and Pál, M. 2005. Stochastic Steiner Trees Without a Root. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, 1051–1063.

Gupta, R.; and Roughgarden, T. 2017. A PAC Approach to Application-Specific Algorithm Selection. *SIAM J. Comput.*, 46(3): 992–1017.

Im, S.; Kumar, R.; Qaem, M. M.; and Purohit, M. 2021. Non-Clairvoyant Scheduling with Predictions. In Agrawal, K.; and Azar, Y., eds., *SPAA '21: 33rd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual Event, USA, 6-8 July, 2021*, 285–294. ACM.

Imase, M.; and Waxman, B. M. 1991. Dynamic Steiner Tree Problem. *SIAM J. Discret. Math.*, 4(3): 369–384.

Indyk, P.; Mallmann-Trenn, F.; Mitrovic, S.; and Rubinfeld, R. 2020. Online Page Migration with ML Advice. *CoRR*, abs/2006.05028.

Jiang, Z.; Panigrahi, D.; and Sun, K. 2020. Online Algorithms for Weighted Paging with Predictions. In Czumaj, A.; Dawar, A.; and Merelli, E., eds., *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, 69:1–69:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Karp, R. M. 1972. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, 85–103.

Kou, L. T.; Markowsky, G.; and Berman, L. 1981. A Fast Algorithm for Steiner Trees. *Acta Informatica*, 15: 141–145.

Kraska, T.; Beutel, A.; Chi, E. H.; Dean, J.; and Polyzotis, N. 2018. The Case for Learned Index Structures. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, 489–504. New York, NY, USA: ACM. ISBN 978-1-4503-4703-7.

Kurz, D.; Mutzel, P.; and Zey, B. 2012. Parameterized Algorithms for Stochastic Steiner Tree Problems. In *Mathematical and Engineering Methods in Computer Science, 8th International Doctoral Workshop, MEMICS 2012, Znojmo, Czech Republic, October 25-28, 2012, Revised Selected Papers*, 143–154.

Lappas, T.; Terzi, E.; Gunopulos, D.; and Mannila, H. 2010. Finding effectors in social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, 1059–1068.

Lattanzi, S.; Lavastida, T.; Moseley, B.; and Vassilvitskii, S. 2020. Online Scheduling via Learned Weights. In Chawla, S., ed., *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, 1859–1877. SIAM.

Lavastida, T.; Moseley, B.; Ravi, R.; and Xu, C. 2021. Learnable and Instance-Robust Predictions for Online Matching, Flows and Load Balancing. In Mutzel, P.; Pagh, R.; and Herman, G., eds., *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, 59:1–59:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Leitner, M.; Ljubic, I.; Luipersbeck, M.; and Sinnl, M. 2018. Decomposition methods for the two-stage stochastic Steiner tree problem. *Comput. Optim. Appl.*, 69(3): 713–752.

Lykouris, T.; and Vassilvtiskii, S. 2018. Competitive Caching with Machine Learned Advice. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 3302–3311. Stockholmsmässan, Stockholm Sweden: PMLR.

Moseley, B.; Vassilvitskii, S.; and Wang, Y. 2021. Hierarchical Clustering in General Metric Spaces using Approximate Nearest Neighbors. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, 2440–2448.

Purohit, M.; Svitkina, Z.; and Kumar, R. 2018. Improving Online Algorithms via ML Predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, 9684–9693.

Rohatgi, D. 2020. Near-Optimal Bounds for Online Caching with Machine Learned Advice. In Chawla, S., ed., *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, 1834–1845. SIAM.

Sadeghi, A.; and Fröhlich, H. 2013. Steiner tree methods for optimal sub-network identification: an empirical study. *BMC Bioinform.*, 14: 144.

Schrijver, A. 2003. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media.

Takakashi, H. 1980. An Approximate Solution for Steiner Problem in Graphs. *Math. Japonica*, 24(6): 573–577.

Wei, A. 2020. Better and Simpler Learning-Augmented Online Caching. *CoRR*, abs/2005.13716.

Westbrook, J. R.; and Yan, D. C. K. 1995. Linear Bounds for On-Line Steiner Problems. *Inf. Process. Lett.*, 55(2): 59–63.

Wu, Y.-F.; Widmayer, P.; and Wong, C.-K. 1986. A faster approximation algorithm for the Steiner problem in graphs. *Acta informatica*, 23(2): 223–229.