Online Steiner Forest with Recourse

Yaowei Long[1]
University of Michigan, Ann Arbor, MI, USA

Sepideh Mahabadi
Microsoft Research, Redmond, WA, USA

Sherry Sarkar
Carnegie Mellon University, Pittsburgh, PA, USA

Jakub Tarnawski
Microsoft Research, Redmond, WA, USA

Abstract

In the online Steiner forest problem we are given a graph G, and a sequence of terminal pairs (u_i, v_i) which arrive in an online fashion. We are asked to maintain a low-cost subgraph in which each u_i is connected to v_i for all the pairs that have arrived so far. If we are not allowed to delete edges from our solution, then the best possible competitive ratio is $\Theta(\log n)$. In this work, we initiate the study of low-recourse algorithms for online Steiner forest. We give an algorithm that maintains a constant-competitive solution and has an amortized recourse of $O(\log n)$, i.e., inserts and deletes $O(\log n)$ edges per demand on average.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online algorithms with recourse, Steiner forest, Network design

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.141

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2605.09821

Acknowledgements The authors want to thank Roie Levin and Anupam Gupta for helpful discussions.

1 Introduction

The **Steiner tree** problem is a classic question in network design. Given a weighted graph and a subset of vertices called *terminals*, it asks to compute the cheapest tree that connects the terminals. Since being included on Karp's list of 21 NP-complete problems [61] it has been extensively studied. The problem is known to be APX-hard [37]; however, getting a constant-factor approximation is straightforward, as computing a minimum spanning tree on the set of terminals already yields a 2-approximation [62]. The current best approximation guarantee is $\ln 4 + \varepsilon < 1.39$ [31].

In the **online** version of the **Steiner tree** problem, introduced in a seminal work of Imase and Waxman [59], the terminals arrive online, and after each arrival the algorithm must produce a solution that connects the current terminal set – without knowledge of the future arrivals and without the possibility of deleting already added edges. They showed that the natural greedy algorithm is $O(\log n)$-competitive, and that no algorithm can obtain a better competitive ratio.

Motivated by this impossibility result, Imase and Waxman [59] also asked the question of whether deleting a small number of edges would allow one to maintain a constant-competitive solution. They showed how to maintain such a Steiner tree while making $O(n^{3/2})$ changes

[1] The work was done while the author was an intern at Microsoft Research.

per n arrivals (which improves upon the $O(n^2)$ changes needed if one naively recomputes the tree after each arrival). In other words, they gave a constant-competitive algorithm with $O(\sqrt{n})$ *amortized recourse*. This was eventually improved over 20 years later by Megow, Skutella, Verschae, and Wiese [68] to constant amortized recourse. Finally, Gu, Gupta, and Kumar [50] showed that it is in fact enough to make just a single edge swap per arrival.

In the more general **Steiner forest** problem, instead of a set of terminals we are given a set of *terminal pairs*, and asked to compute the cheapest forest that connects the vertices of each terminal pair together. The first (offline) approximation algorithm for the problem is a primal-dual approach by Agrawal, Klein, and Ravi [2] that yields a 2-approximation; this was further generalized by Goemans and Williamson [47], and the approximation ratio was improved only very recently to $2 - 2^{-11}$ [3] and subsequently to 1.994 [58]. As for algorithms based on more combinational techniques, a natural greedy algorithm that repeatedly connects the closest yet-unconnected terminal pair is no better than $\Omega(\log n)$ [36], but Gupta and Kumar [55] showed that the *gluttonous algorithm*, which repeatedly connects the two closest terminals (possibly not from the same pair), does yield a constant-factor approximation. Furthermore, Groß, Gupta, Kumar, Matuschke, Schmidt, Schmidt, and Verschae [48] gave a constant-factor approximation based on local search.

The focus of this work is the **online** version of the **Steiner forest** problem, introduced by Westbrook and Yan [79], who gave an $\tilde{O}(\sqrt{n})$-competitive algorithm. Next, Awerbuch, Azar, and Bartal [8] showed that the greedy algorithm is $O(\log^2 n)$-competitive using a primal-dual analysis. Finally, Berman and Coulston [14] gave a primal-dual $O(\log n)$-competitive algorithm. (This is optimal due to the abovementioned hardness result for online Steiner tree [59]. Their algorithm is not greedy; the competitive ratio of greedy is a prominent open problem, see e.g. [10].)

In contrast to Steiner tree, for Steiner forest no $o(\log n)$-competitive, low-recourse algorithm has been known. Finding such an algorithm has been listed as "an interesting open problem" in [54] and "still wide open" in [48].

1.1 Our Results

In this work we give the first low-recourse constant-competitive algorithm for online Steiner forest.

▶ **Theorem 1.** *There is an algorithm for online Steiner forest that, over the course of n arrivals of terminal pairs (u_i, v_i) in a metric space, maintains an $O(1)$-competitive solution while inserting and deleting at most $O(n \log n)$ edges.*

Our approach in fact yields a tradeoff: for any parameter $\lambda \in [1, \log n]$, it achieves competitive ratio $O(\log(n)/\lambda)$ and total recourse $O(n\lambda)$. See Theorem 8 for a detailed version of Theorem 1.

▶ Remark 2. As a byproduct of our approach, we also get an $O(\log n)$-competitive algorithm for the online Steiner forest problem (the classic, no-recourse setting). At a high level, this follows by neglecting to delete the edges that our algorithm decides to delete, and roughly corresponds to simulating the gluttonous algorithm in an online fashion (see Appendix A for a formal algorithm description). This matches the optimal guarantee of Berman and Coulston [14] using a fully combinatorial algorithm as opposed to their primal-dual one.

1.2 Related Work

Online algorithms with low recourse (often called *consistent*) are a very active and growing research area. Such algorithms have been proposed for many problems: clustering and facility location [66, 51, 38, 43, 22, 64, 44, 34, 28], scheduling and load balancing [69, 78, 4, 9, 70, 72, 42, 56, 16, 46, 63, 45], matching [49, 35, 25, 56, 5, 15, 67, 73, 24], matroid intersection [27], set cover [53, 1, 20, 57, 21, 7, 74, 76, 30, 18, 75], graph coloring [77], edge coloring [19], edge orientation [26, 71, 13], online learning [60], maximal independent sets [33, 6, 12], broadcast range assignment [39], spanners [11, 23], discrepancy [52], chasing positive bodies [17], submodular maximization [40, 41, 29], submodular cover [57], or knapsack [32].

Fully dynamic online Steiner tree. Imase and Waxman's $O(\sqrt{n})$-recourse result for Steiner tree [59] in fact also holds in a more general setting where the input sequence consists of terminal insertions as well as deletions. For this setting, Łącki, Oćwieja, Pilipczuk, Sankowski, and Zych [65] gave a constant-competitive algorithm with $O(\log \Delta)$ recourse, where Δ is the ratio of the maximum to minimum distance in the metric. Gupta and Kumar [54] improved this to $O(1)$ recourse. Gupta and Levin [57] recovered the result of [65] in a more general setting of submodular cover. In all of these results, the recourse bounds are amortized.

1.3 Technical Overview

A natural attempt at maintaining an online $O(1)$-approximate Steiner forest is to, for each arrival in the input sequence, independently define the current snapshot of the online solution as the outcome of some offline $O(1)$-approximate algorithm for the current instance. This approach will automatically give an $O(1)$-approximate online solution. To control the recourse, the low-recourse online Steiner tree algorithm [50] exploits a well-behaved offline algorithm which constructs an offline solution under the guidance of a *clustering procedure*.

To design a low-recourse online Steiner forest algorithm, we also start with this framework, and fortunately, a generalized clustering procedure already exists for the Steiner forest problem [55]. However, the following fundamental difference between clustering procedures for Steiner tree and Steiner forest problems prevents us from continuing with the same approach as in [50].

The Barrier. On a high level, a clustering procedure will maintain a clustering $\mathscr{C}$ (i.e., a partition) of the terminals and iteratively merge two *active* clusters in $\mathscr{C}$ which are close enough in the contracted metric $\mathcal{M}/\mathscr{C}$ (we can think of the input metric $\mathcal{M}$ as a complete weighted graph on the terminals[2] and of $\mathcal{M}/\mathscr{C}$ as the graph with each cluster contracted into a single vertex). In the clustering procedure for Steiner tree, clusters are always active, which means that the procedure is equivalent to merging clusters which are close in the original metric $\mathcal{M}$, and therefore, adding one edge to the solution suffices to simulate this merge. In contrast, in the clustering procedure for Steiner forest, some clusters will become inactive (i.e., they will stay in the clustering without participating in further merge operations). Hence, we need to measure the closeness in the contracted metric $\mathcal{M}/\mathscr{C}$, and thus simulating a merge operation between clusters $C_1, C_2 \in \mathscr{C}$ requires us to add into the solution a C_1-C_2 shortest path in $\mathcal{M}/\mathscr{C}$, which potentially consists of many edges.

[2] In our model, we assume that the input metric $\mathcal{M}$ has no Steiner node, i.e., $\mathcal{M}$ is a metric on terminals. See Section 2 and particularly Remark 3 for a further discussion.

We note that the difference between the clustering procedures for these two problems is rooted in the inherent difference between Steiner forest and Steiner tree: a Steiner tree solution must be connected, while a Steiner forest solution may not. This is also the fundamental barrier to overcome when designing Steiner forest algorithms in other settings (e.g., the offline and classic online settings).

In other words, while applying the approach in [50] to Steiner forest may control recourse with respect to merges and their revocations (that is, when comparing the clustering procedures for two consecutive arrivals $t-1$ and t, some merges done at $t-1$ may be revoked, and new merges may be performed at t), it remains unclear how to bound the recourse in terms of edge changes.

Step 1: Pinning Edges. We proceed completely differently from [50] and use the idea of *pinning edges*. On a high level, when performing a merge by buying a shortest path, we *pin* the *cheapest* $(1/\log n)$-fraction of edges on this path, where pinning an edge means that it will not be deleted in the future, even if the corresponding merge is later revoked. To bound the recourse without harming the approximation, we will guarantee the following.

- The number of pinned edges is linear in n at the end, which means that the total recourse will be bounded by $O(n \log n)$. This is relatively simple: intuitively, we will enforce the set of pinned edges to be acyclic.
- The cost of the pinned edges should always be within a constant factor of the optimum, so that we can still guarantee a constant-factor approximation. To this end, an important step is to bound the total cost of all merges throughout the algorithm by $O(\log n \cdot \mathsf{OPT})$, including those that are later revoked (see the following Step 2 for a detailed discussion). This will bound the total cost of pinned edges by $O(\mathsf{OPT})$, since whenever we buy a path, the cheapest $(1/\log n)$-fraction of its edges, which we pin, have cost at most $(1/\log n)$-fraction of the path cost.

Step 2: Bounding the Total Merging Cost. We employ a particular clustering procedure from the offline *timed gluttonous algorithm* [55]. It proceeds by *levels*, from 0 to the top. At each level i, it iteratively merges two clusters if their distance is in $[2^i, 2^{i+1})$ (the merging cost is at most their distance). Furthermore, the clustering procedure allows some flexibility in choosing the order of level-i merges; intuitively, we prioritize the level-i merges that already existed in the previous arrival.

For the above clustering procedure, we can show that for a fixed level i, the cost of all merging operations at this level throughout the algorithm is bounded by $O(\mathsf{OPT})$ using a dual-fitting framework. At a high level, the dual-fitting argument is intuitive: we want to maintain a set of source vertices and obtain a feasible dual solution by growing balls around them. The subtlety in the formal analysis is that choosing (and maintaining) the right source vertices requires exploiting how the offline clustering procedures relate to one another across different arrivals.

Finally, the total merging cost is $O(\log n \cdot \mathsf{OPT})$, since the merges from levels beyond the top $O(\log n)$ will have negligible costs. We emphasize that this total merging cost is not the cost of our online solution. The online solution consists of (i) the output of an offline $O(1)$-approximate algorithm for the current instance, and (ii) the edges pinned up to that point, which have cost $O(\mathsf{OPT})$ as long as the total merging cost is $O(\log n \cdot \mathsf{OPT})$.

1.4 Outline

In Section 2, we introduce the problem setting and notation. In Section 3, we describe an offline clustering procedure from the timed gluttonous algorithm [55], which will be used in our online algorithm. In Section 4, we show our algorithm for maintaining a forest based on the clustering procedure while incurring little recourse. Lastly, we discuss open problems in Section 5.

2 Preliminaries

We use $(\mathcal{M}, D)$ to denote an (offline) Steiner forest instance, where $D = \{(u_i, v_i) \mid 1 \le i \le n\}$ is a set of n demand pairs, and $\mathcal{M} = (V, \mathsf{dist}_{\mathcal{M}})$ is a metric on the *terminals* $V = \{u_i, v_i \mid 1 \le i \le n\}$. We call v_i the *mate* of u_i (and vice versa). We assume $\mathsf{dist}_{\mathcal{M}}$ takes values at least 1, but do not require it to be bounded by a polynomial of n, so this assumption is without loss of generality by scaling. For ease of presentation, we assume that each terminal participates in exactly one demand pair. The more general case, where a terminal may belong to multiple demand pairs, can be handled without additional difficulty[3].

The graph representation of the metric $\mathcal{M}$ is an undirected complete graph (V, E), where each edge $e = (u, v) \in E$ has cost $\mathsf{cost}(e) = \mathsf{dist}_{\mathcal{M}}(u, v)$. Without ambiguity, we use $\mathcal{M}$ to denote this original graph and we call edges in E *original edges* (to differentiate them from *virtual edges* that we introduce later). A feasible solution to the Steiner forest instance $(\mathcal{M}, D)$ is a subset of original edges $F \subseteq E$ such that each demand pair $(u, v) \in D$ has u and v connected in the subgraph (V, F). We do not require a feasible solution F to be acyclic (a forest), but note that an optimal solution must be a forest.

In the online setting, the instance $(\mathcal{M}, D)$ is given in an online fashion. At each moment (called *arrival*) $1 \le t \le n$, a demand pair $(u^{(t)}, v^{(t)}) \in D$ arrives. Moreover, $\mathsf{dist}_{\mathcal{M}}(u^{(t)}, v^{(t)})$ and $\mathsf{dist}_{\mathcal{M}}(u^{(t)}, x), \mathsf{dist}_{\mathcal{M}}(v^{(t)}, x)$ for all arrived terminals x are revealed. In other words, if we let $D^{(t)}$ denote the set of arrived demand pairs and $V^{(t)}$ denote the arrived terminals, then after this arrival, we only know the submetric $\mathcal{M}^{(t)}$ of $\mathcal{M}$ induced by $V^{(t)}$.

For the online instance $\{(\mathcal{M}^{(t)}, D^{(t)}) \mid 1 \le t \le n\}$, an online solution $\{F^{(t)} \mid 1 \le t \le n\}$ has competitive ratio α and amortized recourse δ if each $F^{(t)}$ is an α-approximate solution to the instance $(\mathcal{M}^{(t)}, D^{(t)})$, and the total number of edge insertions and deletions required to maintain the online solution across all n arrivals is at most $n\delta$.

▶ Remark 3. The way we define a Steiner forest instance $(\mathcal{M}, D)$ may seem unusual in the sense that $\mathcal{M}$ is a metric only on the set of terminals (without Steiner vertices). However, we point out that this model is in fact common in the literature on online Steiner trees with low recourse [68, 50, 54]. The motivation for this definition is that maintaining a Steiner tree/forest in a general (non-metric) graph with low recourse is unrealistic, since even purchasing a simple path between two terminals can already incur large recourse.[4] Moreover, restricting to the *terminal-only* metric increases the solution cost by at most a factor of 2 compared to the general model, so this model still captures the Steiner forest problem well, particularly when we aim for an $O(1)$-approximation.

[3] More explicitly, if two terminals s_i and t_j are at the same point v of $\mathcal{M}$, we may replace v with v' and v'' at distance $\varepsilon > 0$, and then scale all distances by $\frac{1}{\varepsilon}$.

[4] In the seminal work [59] on online Steiner trees with recourse, the authors consider a different model in which recourse is measured by the number of *primitive path* insertions and deletions, allowing them to work with general graphs.

▶ Remark 4. We assume that, at the beginning, we know [a constant-estimation $\hat{n}$ of] the length n of the online instance $\{(\mathcal{M}^{(t)}, D^{(t)}) \mid 1 \le t \le n\}$. This assumption can be easily removed as follows. Initially, set the estimation $\hat{n}$ to be a constant. Whenever the current number of arrivals exceeds $\hat{n}$, we double $\hat{n}$ and restart the entire algorithm. Note that these restarts will only increase the final amortized recourse by a constant factor.

Clusterings. Given a terminal set V, a *clustering* $\mathscr{C}$ is a partitioning of V into *clusters*. The *trivial clustering* is the one in which each terminal forms its own singleton cluster. For two clusterings $\mathscr{C}_1$ and $\mathscr{C}_2$ (they can be clusterings of different terminal sets V_1 and V_2), we write $\mathscr{C}_1 \preceq \mathscr{C}_2$ if each cluster $C_1 \in \mathscr{C}_1$ is contained in some cluster $C_2 \in \mathscr{C}_2$, i.e., $C_1 \subseteq C_2$. Note that if $\mathscr{C}_1 \preceq \mathscr{C}_2$, there must be $V_1 \subseteq V_2$.

Contracted Metrics. Given an original metric $\mathcal{M} = (V, \mathsf{dist}_{\mathcal{M}})$ and a clustering $\mathscr{C}$ of V, consider the graph obtained by contracting each cluster $C \in \mathscr{C}$ in the graph representation of the original metric $\mathcal{M}$ into a single vertex representing C. Note that the vertex set of this graph is exactly $\mathscr{C}$. We refer to the shortest-path metric of this graph as $\mathcal{M}/\mathscr{C} = (\mathscr{C}, \mathsf{dist}_{\mathcal{M}/\mathscr{C}})$, and denote this graph by $G_{\text{orig}}(\mathcal{M}/\mathscr{C})$.

Sometimes we will further contract the graph $G_{\text{orig}}(\mathcal{M}/\mathscr{C})$ by a subset $E' \subseteq E$ of original edges. That is, consider edges $e \in E'$ one by one, and for each $e = (u, v)$, contract the two vertices corresponding to u, v into a single vertex (it is possible that u, v correspond to the same vertex in the current graph, in which case we do nothing). We denote the resulting graph by $G_{\text{orig}}((\mathcal{M}/\mathscr{C})/E')$, and use $\mathsf{dist}_{(\mathcal{M}/\mathscr{C})/E'}(\cdot,\cdot)$ to denote its shortest-path metric. The vertex set of $G_{\text{orig}}((\mathcal{M}/\mathscr{C})/E')$ naturally corresponds to a partition of the vertex set of $G_{\text{orig}}(\mathcal{M}/\mathscr{C})$ (which is exactly $\mathscr{C}$), so for $C_1, C_2 \in \mathscr{C}$, $\mathsf{dist}_{(\mathcal{M}/\mathscr{C})/E'}(C_1, C_2)$ can be defined naturally and it is unambiguous to talk about a C_1-C_2 shortest path in $G_{\text{orig}}((\mathcal{M}/\mathscr{C})/E')$.

3 A Clustering Procedure

In this section, we describe a clustering procedure which is the core part of the offline constant-approximate Steiner forest algorithm (called the *timed gluttonous algorithm*) in [55].

The input of the clustering procedure is a Steiner forest instance $(\mathcal{M}, D)$, and it outputs a *clustering hierarchy* $\mathscr{H} = (\mathscr{C}_0, \mathscr{C}_1, \mathscr{C}_2, ..., \mathscr{C}_{L+1})$ of the terminal set V, where L denotes the maximum level (which will be defined shortly). The hierarchy $\mathscr{H}$ satisfies that $\mathscr{C}_0$ is the trivial clustering of V, and for each $0 \le i \le L$, $\mathscr{C}_i \preceq \mathscr{C}_{i+1}$.

The Clustering Procedure. Initially, for each terminal $v \in V$, define its *level* to be

$$\mathsf{level}(v) = \lceil \log_2 \mathsf{dist}_{\mathcal{M}}(v, u) \rceil$$

where u is the mate of v. Note that $\mathsf{level}(v) \ge 0$ since we have assumed $\mathsf{dist}_{\mathcal{M}}(\cdot,\cdot)$ takes values at least 1. The level of a cluster $C \subseteq V$ is $\mathsf{level}(C) = \max_{v \in C} \mathsf{level}(v)$. Let $L = \max_{v \in V} \mathsf{level}(v)$ be the maximum level.

Then we iterate i from 0 to L. For each iteration i, we will construct $\mathscr{C}_{i+1}$ based on $\mathscr{C}_i$ as follows.

1. First, we construct an auxiliary graph H_i, called the *virtual graph*, with vertices

$$V(H_i) = \{C \in \mathscr{C}_i \mid \mathsf{level}(C) \ge i\}$$

corresponding to clusters $C \in \mathscr{C}_i$ with $\mathsf{level}(C) \geq i$, called *i-active* clusters. Naturally, each $C \in \mathscr{C}_i$ with $\mathsf{level}(C) < i$ is an *i-inactive* cluster. For each $C_1, C_2 \in V(H_i)$, there is an edge (called a *virtual edge*) in H_i connecting them iff

$$\mathsf{dist}_{\mathcal{M}/\mathscr{C}_i}(C_1, C_2) < 2^{i+1}.$$

2. We construct $\mathscr{C}_{i+1}$ by *contracting H_i over $\mathscr{C}_i$*. That is, we first copy all i-inactive clusters from $\mathscr{C}_i$ into $\mathscr{C}_{i+1}$. Next, for each connected component Q of H_i, we add a cluster C_{i+1} into $\mathscr{C}_{i+1}$ which is the union of all $\mathscr{C}_i$-clusters in this connected component, i.e., $C_{i+1} = \bigcup_{C_i \in Q} C_i$ (note that the C_{i+1} added in this way are i-active). Note that $\mathscr{C}_i \preceq \mathscr{C}_{i+1}$ by the construction.

We note that the above clustering procedure is identical to that in Section 4.1 of [55], so we list some observations below without formal proofs. These observations will be useful when analyzing our online algorithm in Section 4.

▶ **Observation 5.** *For each $i \geq 0$ and any two different i-active clusters $C_1, C_2 \in \mathscr{C}_i$, we have $\mathsf{dist}_{\mathcal{M}/\mathscr{C}_i}(C_1, C_2) \geq 2^i$.*

The above Observation 5 follows because in iteration $i-1$, we merge two $(i-1)$-active clusters in $\mathscr{C}_{i-1}$ if they are close (i.e., they have distance less than 2^i in $\mathcal{M}/\mathscr{C}_{i-1}$). A formal proof can be found in Section 4 in [55].

▶ **Observation 6.** *For each demand pair $(u, v) \in D$, there is a cluster $C \in \mathscr{C}_{L+1}$ such that $u, v \in C$.*

This is basically because $\mathsf{dist}_{\mathcal{M}}(u, v) < 2^{\mathsf{level}(u)+1}$ by the definitions of $\mathsf{level}(\cdot)$ and L. Then at the beginning of iteration $i = \mathsf{level}(u)$, if u and v are still inside different clusters $C_u, C_v \in \mathscr{C}_i$, then C_u and C_v are still i-active and they will be merged in this iteration.

The Forest-Forming Procedure. For better understanding, we briefly describe a procedure forming a Steiner forest solution based on the clustering hierarchy (which is actually straightforward). However, we are not going to formally show the feasibility and approximation of this procedure, since the forest-forming procedure in our online algorithm is specialized.

We initialize an empty solution F. For each i from 0 to L, we select an arbitrary *virtual spanning forest* $\hat{F}_i$ of the virtual graph H_i, and then for each virtual edge $\hat{e} \in \hat{F}_i$ connecting two clusters $C_1, C_2 \in \mathscr{C}_i$, add into F the original edges on a C_1-C_2 shortest path P in the graph $G_{\text{orig}}(\mathcal{M}/\mathscr{C}_i)$. Note that the path P has cost at most 2^{i+1}, as guaranteed by the definition of virtual edges in Step 1.

The feasibility of the solution F basically comes from Observation 6, and the approximation is given by the following Lemma 7, which will also be useful when analyzing the approximation of our online algorithm in Section 4. Lemma 7 is proved almost explicitly[5] in the proof of Theorem 4.2 in [55].

▶ **Lemma 7** ([55]). *Let OPT be the cost of an optimal solution to the instance $(\mathcal{M}, D)$. Then*

$$\sum_{i=0}^{L} (|\mathscr{C}_i| - |\mathscr{C}_{i+1}|) \cdot 2^{i+1} \leq O(1) \cdot \mathsf{OPT}\,.$$

[5] The m_i in their proof is exactly $|\hat{F}_i|$ in our context. They show that $\sum_i m_i 2^{i+1} \leq O(1) \cdot \mathsf{OPT}$. Combining this with $|\hat{F}_i| = |\mathscr{C}_i| - |\mathscr{C}_{i+1}|$ gives Lemma 7.

Let us explain Lemma 7 somewhat further. The left-hand side takes a summation over all levels i. For each level i, observe that $|\mathscr{C}_i| - |\mathscr{C}_{i+1}| = |\hat{F}_i|$. Hence the inequality basically says that if we assign a budget of 2^{i+1} to each virtual edge in $\hat{F}_i$, then the total budget is within a constant factor of OPT. Indeed, the cost of the original edges added into the real solution F by each virtual edge is within the budget.

4 Our Online Algorithm

In this section, we will present our online Steiner forest algorithm with low recourse, and prove Theorem 8.

▶ **Theorem 8.** *There is an algorithm for online Steiner forest that, over the course of n arrivals of terminal pairs (u_i, v_i) in a metric space, maintains an $O(\log(n)/\lambda)$-competitive solution while inserting and deleting at most $O(n\lambda)$ edges, for any $\lambda \in [1, \log n]$.*

Throughout this section, λ is the tradeoff parameter in Theorem 8, and Theorem 1 follows Theorem 8 by setting $\lambda = \log n$.

We start with Section 4.1, in which we will apply the clustering procedure from Section 3 for each arrival and introduce some notation and observations regarding the clustering hierarchies. Next, in Section 4.2, we define the online Steiner forest solution F algorithmically by, for each arrival t, constructing the *snapshot* $F^{(t)}$ of F based on the current clustering hierarchy. Finally, Theorem 8 follows from the analyses of feasibility, recourse, and approximation in Section 4.3, Section 4.4, and Section 4.5, respectively.

For convenience, we provide an outline of the full online algorithm as Algorithm 2 (page 11). The details of each step are still described in the text of Section 4.2.

4.1 Applying the Clustering Procedure

We start with some notation. Let $\{(\mathcal{M}^{(t)}, D^{(t)}) \mid 1 \le t \le n\}$ be the given online instance. For each arrival t, we run the clustering procedure for the (offline) instance $(\mathcal{M}^{(t)}, D^{(t)})$. Here we will write the corresponding variables L (the maximum level), H_i (the virtual graph at level i), $\mathscr{C}_i$ (the clustering at level i), and $\mathscr{H}$ (the clustering hierarchy) with a superscript (t). To avoid clutter, for each arrival $t \ge 1$ and $i \ge L^{(t)} + 1$, we let $\mathscr{C}_i^{(t)}$ be the same as the top-level clustering $\mathscr{C}_{L^{(t)}+1}^{(t)}$, and let $H_i^{(t)}$ be an empty virtual graph. Furthermore, for $t = 0$, we define $L^{(0)} = 0$ and each $\mathscr{C}_i^{(0)}$ and $H_i^{(0)}$ to be empty. The Lemma 9 below will be useful later.

▶ **Lemma 9.** *For each $t \ge 1$ and $0 \le i \le L^{(t)} + 1$, we have $\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_i^{(t)}$.*

Proof. We prove this by induction on i. Since $\mathscr{C}_0^{(t-1)}$ is the trivial clustering on the terminal set $V^{(t-1)}$ and $\mathscr{C}_0^{(t)}$ is the trivial clustering on the terminal set $V^{(t)}$, the base case is true. Now assume for some fixed i, $\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_i^{(t)}$. We will show $\mathscr{C}_{i+1}^{(t-1)} \preceq \mathscr{C}_{i+1}^{(t)}$. In other words, our goal is to show that a cluster $C \in \mathscr{C}_{i+1}^{(t-1)}$ is fully contained in some cluster of $\mathscr{C}_{i+1}^{(t)}$. To this end, we consider the connected components of $H_i^{(t-1)}$ and the connected components of $H_i^{(t)}$. Note that if some two clusters $C_1, C_2 \in \mathscr{C}_i^{(t-1)}$ have

$$\mathsf{dist}_{\mathcal{M}/\mathscr{C}_i^{(t-1)}}(C_1, C_2) < 2^{i+1}$$

then, since $\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_i^{(t)}$, we also have that for the parent clusters $D_1, D_2 \in \mathscr{C}_i^{(t)}$ which contain C_1 and C_2 respectively,

$$\mathsf{dist}_{\mathcal{M}/\mathscr{C}_i^{(t)}}(D_1, D_2) < 2^{i+1}.$$

Therefore, an edge between C_1 and C_2 in $H_i^{(t-1)}$ is also an edge between D_1 and D_2 in $H_i^{(t)}$. So, if a cluster $C \in \mathscr{C}_{i+1}^{(t-1)}$ results from some connected component Q in $H_i^{(t-1)}$ consisting of $C_1, \ldots, C_k$, then there is a corresponding connected component Q' of $D_1, \ldots, D_k$ in $H_i^{(t)}$ (note that the D_ℓ are not necessarily distinct). This proves our inductive step. ◀

4.2 Constructing the Snapshots of the Online Solution

Consider an arrival $1 \le t \le n$. We construct the snapshot $F^{(t)}$ of the online solution in two phases.

The First Phase. We first construct a *virtual solution* $\hat{F}^{(t)}$ using virtual edges in the virtual graphs $H_i^{(t)}$ for all i. Similarly to the forest-forming procedure in Section 3, for each level $0 \le i \le L^{(t)}$, we will pick a virtual spanning forest $\hat{F}_i^{(t)}$ of $H_i^{(t)}$. However, rather than choosing an arbitrary forest as $\hat{F}_i^{(t)}$, we will choose $\hat{F}_i^{(t)}$ more carefully, as we discuss shortly. The virtual solution $\hat{F}^{(t)} = \bigcup_{0 \le i \le L^{(i)}} \hat{F}_i^{(t)}$ is simply the union of virtual spanning forests at all levels.

Inheritable and Inherited Virtual Edges. Before describing how to choose $\hat{F}_t^{(i)}$, we need to introduce the concepts of *inheritable/non-inheritable* and *inherited/non-inherited* virtual edges.

For each virtual edge $\hat{e}^{(t-1)} = (C_1, C_2) \in \hat{F}_i^{(t-1)}$, it is *inheritable* if $C_1, C_2 \in \mathscr{C}_i^{(t-1)}$ are not contained in the same $\mathscr{C}_i^{(t)}$-cluster (recall that $\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_i^{(t)}$ from Lemma 9), otherwise it is *non-inheritable*.

▶ **Lemma 10.** *For each inheritable virtual edge $\hat{e}^{(t-1)} = (C_1, C_2) \in \hat{F}_i^{(t-1)}$, there is a virtual edge $\hat{e}^{(t)} \in H_i^{(t)}$ connecting $D_1, D_2 \in \mathscr{C}_i^{(t)}$, the two different $\mathscr{C}_i^{(t)}$-clusters containing C_1, C_2 respectively.*

Proof. In the same vein as the proof above, note that if $\hat{e}^{(t-1)} = (C_1, C_2)$ is an inheritable virtual edge in $H_i^{(t-1)}$, then: (1) $C_1, C_2 \in \mathscr{C}_i^{(t-1)}$ are contained in distinct clusters D_1 and D_2 in $\mathscr{C}_i^{(t)}$, and (2) $\mathsf{dist}_{\mathcal{M}/\mathscr{C}_i^{(t-1)}}(C_1, C_2) < 2^{i+1}$. The second property implies $\mathsf{dist}_{\mathcal{M}/\mathscr{C}_i^{(t)}}(D_1, D_2) < 2^{i+1}$, and therefore we will have an edge $\hat{e}^{(t)} \in H_i^{(t)}$ connecting D_1 and D_2. ◀

Lemma 10 naturally defines a mapping $\pi_i^{(t)}$ from the inheritable virtual edges in $\hat{F}_i^{(t-1)}$ to virtual edges in $H_i^{(t)}$. We define the *inherited* virtual edges in $H_i^{(t)}$ to be the image set of $\pi_i^{(t)}$, and will call the other virtual edges in $H_i^{(t)}$ *non-inherited*. For each inherited virtual edge $\hat{e}^{(t)} \in H_i^{(t)}$, we fix an arbitrary inheritable virtual edge $\hat{e}^{(t-1)} \in \hat{F}_i^{(t-1)}$ with $\pi_i^{(t)}(\hat{e}^{(t-1)}) = \hat{e}^{(t)}$ as the *parent* of $\hat{e}^{(t)}$, and say that $\hat{e}^{(t)}$ is inherited from its parent $\hat{e}^{(t-1)}$.

Choosing $\hat{F}_t^{(i)}$. Now, we pick the virtual spanning forest $\hat{F}_i^{(t)}$ giving priority to the inherited virtual edges in $H_i^{(t)}$. Formally speaking, we first pick an arbitrary spanning forest $\hat{F}_{\mathrm{inh},i}^{(t)}$ in $H_{\mathrm{inh},i}^{(t)}$, the subgraph of $H_i^{(t)}$ induced by the inherited virtual edges. Then we arbitrarily augment $\hat{F}_{\mathrm{inh},i}^{(t)}$ to be a spanning forest $\hat{F}_i^{(t)}$ of $H_i^{(t)}$.

For better understanding, we emphasize that virtual edges in $\hat{F}_i^{(t)}$ are classified in two ways: inheritable vs. non-inheritable, and inherited ($\hat{F}_{\mathrm{inh},i}^{(t)}$) vs. non-inherited ($\hat{F}_i^{(t)} \setminus \hat{F}_{\mathrm{inh},i}^{(t)}$). Each inherited virtual edge in $\hat{F}_i^{(t)}$ is inherited from its parent, some inheritable virtual edge in $\hat{F}_i^{(t-1)}$. However, not every inheritable virtual edge in $\hat{F}_i^{(t-1)}$ may serve as the parent of an inherited virtual edge in $\hat{F}_i^{(t)}$.

Note that $\mathscr{C}_{i+1}^{(t)}$ is exactly the clustering obtained by contracting $\hat{F}_i^{(t)}$ over $\mathscr{C}_i^{(t)}$ (the meaning of contracting is the same as in Step 2 of the clustering procedure), and we let $\mathscr{C}_{\mathrm{inh},i}^{(t)}$ denote the clustering obtained by contracting $\hat{F}_{\mathrm{inh},i}$ over $\mathscr{C}_i^{(t)}$. Then we have the following Lemma 11 which will be useful later.

▶ **Lemma 11.** $\mathscr{C}_{i+1}^{(t-1)} \preceq \mathscr{C}_{\mathrm{inh},i}^{(t)}$.

Proof. Assume for contradiction that there exists a cluster $C_{i+1}^{(t-1)} \in \mathscr{C}_{i+1}^{(t-1)}$ such that $C_{i+1}^{(t-1)}$ intersects two different clusters in $\mathscr{C}_{\mathrm{inh},i}^{(t)}$.

First, we claim that $C_{i+1}^{(t-1)}$ is i-active. Assume the opposite. We must have $C_{i+1}^{(t-1)} \in \mathscr{C}_i^{(t-1)}$ by the description of the clustering procedure. By Lemma 9, we know

$$\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_i^{(t)} \preceq \mathscr{C}_{\mathrm{inh},i}^{(t)}.$$

Hence $C_{i+1}^{(t-1)}$ is contained in some cluster in $\mathscr{C}_{\mathrm{inh},i}^{(t)}$, a contradiction.

Given that $C_{i+1}^{(t-1)}$ is i-active, it is the union of several i-active clusters in $\mathscr{C}_i^{(t-1)}$ by the clustering procedure. Then, by $\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_{\mathrm{inh},i}^{(t)}$ and the assumption that $C_{i+1}^{(t-1)}$ intersects two different clusters in $\mathscr{C}_{\mathrm{inh},i}^{(t)}$, there must be two different i-active $\mathscr{C}_i^{(t-1)}$-clusters $C_{i,1}^{(t-1)}$ and $C_{i,2}^{(t-1)}$ inside $C_{i+1}^{(t-1)}$ such that they are contained by different $\mathscr{C}_{\mathrm{inh},i}^{(t)}$-clusters.

Let $C_{i,1}^{(t)}$ and $C_{i,2}^{(t)}$ be the $\mathscr{C}_i^{(t)}$-clusters containing $C_{i,1}^{(t-1)}$ and $C_{i,2}^{(t-1)}$ respectively (we use $\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_i^{(t)}$ from Lemma 9 again). If $C_{i,1}^{(t)}$ and $C_{i,2}^{(t)}$ are the same, combining it with $\mathscr{C}_i^{(t)} \preceq \mathscr{C}_{\mathrm{inh},i}^{(t)}$ already leads to a contradiction. Hence we assume $C_{i,1}^{(t)}$ and $C_{i,2}^{(t)}$ are different clusters. Now, we have already known that $C_{i,1}^{(t-1)}$ and $C_{i,2}^{(t-1)}$ are in the same connected component of $H_i^{(t-1)}$ (i.e., they are connected by the virtual spanning forest $\hat{F}_i^{(t-1)}$). From the inheritance relation between $\hat{F}_i^{(t-1)}$-virtual edges and $H_i^{(t)}$-virtual edges, it is straightforward to see that $C_{i,1}^{(t)}$ and $C_{i,2}^{(t)}$ are in the same connected component of $H_{\mathrm{inh},i}^{(t)}$, which means they are contained by the same $\mathscr{C}_{\mathrm{inh},i}^{(t)}$-cluster, a contradiction. ◀

The Second Phase. In the second phase, we will construct the snapshot $F^{(t)}$ based on the virtual solution $\hat{F}^{(t)}$. A natural attempt is to proceed analogously to the offline forest-forming procedure in Section 3: for each virtual edge $\hat{e} = (C_1, C_2) \in \hat{F}_i^{(t)} \subseteq \hat{F}^{(t)}$, add into $F^{(t)}$ a C_1-C_2 shortest path in the metric $\mathcal{M}/\mathscr{C}_i^{(t)}$. This will give a feasible $F^{(t)}$ with good approximation as we discussed in Section 3. However, it gives no control over the recourse: such a shortest path may be made up of many original edges in the original metric $\mathcal{M}$.

We fix this issue by *pinning* edges. Roughly speaking, whenever we add a shortest path, we pin some fraction of its original edges. Pinning an original edge means that we will never remove it from our online solution. We will guarantee two properties of the pinned edges. First, the total cost of pinned edges is always within a constant factor of the optimum, so that the online solution can achieve a constant competitive ratio. Second, the number of pinned edges grows linearly in t, so that we can achieve low recourse by charging the number of original edge insertions to the number of pinned edges. We now formalize this argument.

As demands arrive, we will maintain a growing set A of pinned edges. Our job in the current arrival t is to assign to each virtual edge $\hat{e}^{(t)} \in \hat{F}^{(t)}$ a set of original edges in $\mathcal{M}$, denoted by $E_{\mathrm{orig}}(\hat{e}^{(t)})$. From the previous arrival $t-1$, we are already given the pinned edge set $A^{(t-1)}$ and the original edge set $E_{\mathrm{orig}}(\hat{e}^{(t-1)})$ of every virtual edge $\hat{e}^{(t-1)} \in \hat{F}^{(t-1)}$. Meanwhile, we will also update the pinned edge set from $A^{(t-1)}$ to $A^{(t)}$. The original solution is then

$$F^{(t)} = A^{(t)} \cup \bigcup_{\hat{e}^{(t)} \in \hat{F}^{(t)}} E_{\mathrm{orig}}(\hat{e}^{(t)}).$$

For each inherited virtual edge $\hat{e}^{(t)} \in \hat{F}^{(t)}$, its original edge set is simply

$$E_{\text{orig}}(\hat{e}^{(t)}) = E_{\text{orig}}(\hat{e}^{(t-1)})$$

where $\hat{e}^{(t-1)} \in \hat{F}^{(t-1)}$ is the parent of $\hat{e}^{(t)}$. As for pinning, on a high level, we add a $1/\lambda$ fraction of the original edges to $A^{(t)}$ (selecting the cheapest ones); if there are fewer than λ many, we use a buffer set. More precisely, the original edge sets of the non-inherited virtual edges and the update from $A^{(t-1)}$ to $A^{(t)}$ are given by the following algorithm.

Algorithm 1 Pinning.

Initialize $A^{(t)} \leftarrow A^{(t-1)}$
Initialize $B \leftarrow \emptyset$ to be a *multiset* ▷ initialize the buffer
for each level i and non-inherited $\hat{e}^{(t)} = (C_1, C_2) \in \hat{F}_i^{(t)}$ **do**
    $P \leftarrow$ a C_1-C_2 shortest path in $G_{\text{orig}}((\mathcal{M}/\mathscr{C}_i^{(t)})/A^{(t)})$
    $E_{\text{orig}}(\hat{e}^{(t)}) \leftarrow$ the original edges on P
    if $|E_{\text{orig}}(\hat{e}^{(t)})| \geq \lambda$ **then**
        Add $\lfloor |E_{\text{orig}}(\hat{e}^{(t)})|/\lambda \rfloor$ cheapest edges from $E_{\text{orig}}(\hat{e}^{(t)})$ into $A^{(t)}$
        $B \leftarrow \emptyset$
    else
        $B \leftarrow B \uplus E_{\text{orig}}(\hat{e}^{(t)})$ ▷ multiset union
        if $|B| \geq \lambda$ **then**
            Add the cheapest edge from B into $A^{(t)}$
            $B \leftarrow \emptyset$
        end if
    end if
end for

We note that we define the buffer B to be a multiset just for ease of analysis. Defining B as a set instead will not affect the correctness.

Finally, to enhance understanding, we outline our entire online algorithm in Algorithm 2.

Algorithm 2 Online Low-Recourse Steiner Forest.

for each arrival t with demand pair $(u^{(t)}, v^{(t)})$ **do**
    Compute the clustering hierarchy $\mathscr{H}^{(t)} = \{\mathscr{C}_0^{(t)}, \mathscr{C}_1^{(t)}, \mathscr{C}_1^{(t)}, ...\}$ ▷ see Section 3
    for each level i **do**
        Compute $\hat{F}_i^{(t)}$, a set of virtual edges (between clusters in $\mathscr{C}_i^{(t)}$), each of which is either inherited or non-inherited ▷ the first phase
    end for
    The virtual solution is $\hat{F}^{(t)} := \bigcup_i \hat{F}_i^{(t)}$
    Update the pinned edge set $A^{(t)}$ and obtain the mapping E_{orig} from virtual edges $\hat{F}^{(t)}$ to sets of original edges, via Algorithm 1 ▷ the second phase
    Output online solution $F^{(t)} := A^{(t)} \cup \bigcup_{\hat{e}^{(t)} \in \hat{F}^{(t)}} E_{\text{orig}}(\hat{e}^{(t)})$
end for

4.3 Feasibility

To prove feasibility, it suffices to show that virtual edges between two clusters C_1 and C_2 do indeed correspond to real edge sets which connect C_1 and C_2.

▶ **Lemma 12.** *For each arrival t, level $0 \le i \le L^{(t)}$, and virtual edge $\hat{e}^{(t)} = (C_1^{(t)}, C_2^{(t)}) \in \hat{F}_i^{(t)}$, the clusters $C_1^{(t)}$ and $C_2^{(t)}$ belong to the same vertex in the graph $G_{\text{orig}}((\mathcal{M}/\mathscr{C}_i^{(t)})/(A^{(t)} \cup E_{\text{orig}}(\hat{e}^{(t)})))$.*

Proof. If $\hat{e}^{(t)}$ is a non-inherited virtual edge, this statement directly follows from Lines 4 and 5 of Algorithm 1. Concretely, at the moment E_{orig} is defined at Line 5, $C_1^{(t)}$ and $C_2^{(t)}$ clearly belong to the same vertex in $(\mathcal{M}/\mathscr{C}_i^{(t)})/(A^{(t)} \cup E_{\text{orig}}(\hat{e}^{(t)}))$ by definition. Therefore, at the end of arrival t, since $A^{(t)}$ only grows during Algorithm 1, the statement also holds.

The argument for an inherited virtual edge $\hat{e}^{(t)}$ is similar, but we need induction to formally prove it. Assume inductively that the statement of Lemma 12 holds for all virtual edges in $\hat{F}^{(t-1)}$. Let $\hat{e}^{(t-1)} = (C_1^{(t-1)}, C_2^{(t-1)}) \in \hat{F}_i^{(t-1)}$ be the parent of $\hat{e}^{(t)}$, which means $C_1^{(t)} \supseteq C_1^{(t-1)}$, $C_2^{(t)} \supseteq C_2^{(t-1)}$ and $E_{\text{orig}}(\hat{e}^{(t)}) = E_{\text{orig}}(\hat{e}^{(t-1)})$. Then the statement holds for $\hat{e}^{(t)}$ by the induction hypothesis and because $\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_i^{(t)}$ ◀

And so, since $\hat{F}_i^{(t)}$ virtually connects up demands in the arrived demand set (by Observation 6), we know that $F_i^{(t)}$ will be a feasible solution.

4.4 Recourse

We are going to bound the total number of edge insertions by $O(n\lambda)$, that is,

$$\sum_{t=1}^{n} |F^{(t)} \setminus F^{(t-1)}| \le O(n\lambda),$$

which gives $O(\lambda)$ amortized recourse (since the number of edge deletions does not exceed the number of edge insertions).

Consider the changes from $F^{(t-1)}$ to $F^{(t)}$. The new edges $F^{(t)} \setminus F^{(t-1)}$ come from either $A^{(t)} \setminus A^{(t-1)}$ or $\bigcup_{\text{non-inherited } \hat{e}^{(t)} \in \hat{F}^{(t)}} E_{\text{orig}}(\hat{e}^{(t)})$ (since each inherited $\hat{e}^{(t)} \in \hat{F}^{(t)}$ has $E_{\text{orig}}(\hat{e}^{(t)}) \subseteq \hat{F}^{(t-1)}$). In other words, we have

$$\sum_{t=1}^{n} |F^{(t)} \setminus F^{(t-1)}| \le \sum_{t=1}^{n} \left(|A^{(t)} \setminus A^{(t-1)}| + \sum_{\text{non-inherited } \hat{e}^{(t)} \in \hat{F}^{(t)}} |E_{\text{orig}}(\hat{e}^{(t)})| \right).$$

For the pinned edge set, we have $\sum_{t=1}^{n} |A^{(t)} \setminus A^{(t-1)}| \le 2n - 1$ by the fact that A only grows and by the following Lemma 13.

▶ **Lemma 13.** *There are at most $2n - 1$ pinned edges, i.e., $|A^{(n)}| \le 2n - 1$.*

Proof. Observe that whenever we pin several edges at Line 7 or one edge at Line 12, the new pinned edges do not form a cycle with the existing pinned edges on the terminal set $V^{(t)}$. Hence at the end $A^{(n)}$ is a forest on $V^{(n)}$, which implies that $|A^{(n)}| \le |V^{(n)}| - 1 = 2n - 1$. ◀

Next, we charge the total size of original edge sets of non-inherited virtual edges (*non-inherited original edge sets* for short) to the number of pinned edges. In every execution of Line 7, we charge each new pinned edge a fee of 10λ units to account for $|B| + |E_{\text{orig}}(\hat{e}^{(t)})|$, which is feasible because

$$|B| + |E_{\text{orig}}(\hat{e}^{(t)})| \le 10\lambda \cdot \lfloor |E_{\text{orig}}(\hat{e}^{(t)})|/\lambda \rfloor$$

when $|E_{\text{orig}}(\hat{e}^{(t)})| \geq \lambda$ and $|B| < \lambda$ (guaranteed by Algorithm 1). In every execution of Line 12, again charge the new pinned edge a fee of 10λ units to account for $|B|$, which is available because $|B| \leq 2\lambda$ (right before Line 10, the buffer B has size less than λ, and the current $E_{\text{orig}}(\hat{e}^{(t)})$ has size less than λ). Finally, observe that every non-inherited original edge set is accounted for in the charging argument (either directly or via the buffer B), except for those remaining in the buffer B at the end of each arrival (at these moments, $|B| < \lambda$). Therefore, we have

$$\sum_{t=1}^{n} \sum_{\text{non-inherited } \hat{e}^{(t)} \in \hat{F}^{(t)}} |E_{\text{orig}}(\hat{e}^{(t)})| \leq 10\lambda \cdot |A^{(n)}| + n\lambda \leq O(n\lambda).$$

4.5 Approximation

To avoid clutter, we only show that after the last arrival n, the online solution $F^{(n)}$ is an $O(1)$-approximation to the instance $(\mathcal{M}^{(n)}, D^{(n)})$. The approximation in the other arrivals can be bounded in exactly the same way.

Let OPT be the cost of the optimal solution to $(\mathcal{M}^{(n)}, D^{(n)})$. Recall that

$$F^{(n)} = A^{(n)} \cup \bigcup_{\hat{e}^{(n)} \in \hat{F}^{(n)}} E_{\text{orig}}(\hat{e}^{(n)}).$$

Hence, to bound the cost of $F^{(n)}$ by $O(\log n \cdot \mathsf{OPT}/\lambda)$, it suffices to bound

- $\mathsf{cost}(A^{(n)})$ (called the *pinning cost*) by $O(\log n \cdot \mathsf{OPT}/\lambda)$, and
- $\sum_{\hat{e}^{(n)} \in \hat{F}^{(n)}} \mathsf{cost}(E_{\text{orig}}(\hat{e}^{(n)}))$ (called the *forest-forming cost*) by $O(\mathsf{OPT})$.

The Forest-Forming Cost. The forest-forming cost can be interpreted as the cost of the offline Steiner forest algorithm introduced in Section 3, which can be directly bounded by Lemma 7.

Formally, for each level i and each virtual edge $\hat{e}^{(n)} \in \hat{F}_i^{(n)}$, we have

$$\mathsf{cost}(E_{\text{orig}}(\hat{e}^{(n)})) \leq 2^{i+1}. \tag{1}$$

To see this, we trace $\hat{e}^{(n)}$ back through the inheritance relation until reaching a non-inherited virtual edge $\hat{e}^{(t)} = (C_1, C_2) \in \hat{F}_i^{(t)}$ in some arrival $t \leq n$. At the moment when $E_{\text{orig}}(\hat{e}^{(t)})$ is defined on Lines 4 and 5 of Algorithm 1, we have

$$\mathsf{cost}(E_{\text{orig}}(\hat{e}^{(t)})) \leq \mathsf{dist}_{(\mathcal{M}/\mathscr{C}_i^{(t)}) \setminus A^{(t)}}(C_1, C_2) \leq \mathsf{dist}_{\mathcal{M}/\mathscr{C}_i^{(t)}}(C_1, C_2) \leq 2^{i+1}, \tag{2}$$

where the last inequality is by $\hat{F}_i^{(t)} \subseteq H_i^{(t)}$ and the definition of $H_i^{(t)}$. The inheritance relation implies $E_{\text{orig}}(\hat{e}^{(n)}) = E_{\text{orig}}(\hat{e}^{(t)})$, so we get the desired bound (1).

Given (1), we bound the forest-forming cost by

$$\sum_{\hat{e}^{(n)} \in \hat{F}^{(n)}} \mathsf{cost}(E_{\text{orig}}(\hat{e}^{(n)})) \leq \sum_i |\hat{F}_i^{(n)}| \cdot 2^{i+1} = \sum_i (|\mathscr{C}_i^{(n)}| - |\mathscr{C}_{i+1}^{(n)}|) \cdot 2^{i+1} \leq O(\mathsf{OPT}),$$

where the equality is by $|\hat{F}_i^{(n)}| = |\mathscr{C}_i^{(n)}| - |\mathscr{C}_{i+1}^{(n)}|$ and the last inequality is by Lemma 7.

The Pinning Cost. Algorithm 1 pins roughly a $(1/\lambda)$-fraction of edges from the non-inherited original edge sets, so naturally the first step is to bound the total cost of non-inherited original edge sets.

▶ **Theorem 14.** *For each level i we have*

$$\sum_t |\hat{F}_i^{(t)} \setminus \hat{F}_{\mathrm{inh},i}^{(t)}| \cdot 2^{i+1} \le O(\mathsf{OPT})\,. \tag{3}$$

We say that a non-inherited original edge set is from level i if its corresponding non-inherited virtual edge is from $\hat{F}_i^{(t)}$ for some t. Theorem 14 above shows that for each level i, the total cost of non-inherited original edge sets summed over all arrivals is bounded by $O(\mathsf{OPT})$ (note that each non-inherited original edge set has cost at most 2^{i+1} as we argued in (2)).

We prove Theorem 14 in Section 4.6 below. Given Theorem 14, the intuition for bounding the pinning cost is as follows. If we assume that there are $O(\log n)$ levels, then Theorem 14 shows that the total cost of non-inherited original edge sets is $O(\log n \cdot \mathsf{OPT})$, so the pinning cost can be bounded by $O(\log n \cdot \mathsf{OPT}/\lambda)$ since we pinned roughly a $(1/\lambda)$ fraction. Without the assumption of $O(\log n)$ levels, the intuition is that non-inherited original edge sets not from the top $O(\log n)$ levels have negligible costs. We formalize this intuition as follows.

To see $\mathsf{cost}(A^{(n)}) \le O(\mathsf{OPT})$, by Lemma 13 it suffices to show $\mathsf{cost}(A') \le O(\mathsf{OPT})$, where $A' = \{e \mid e \in A^{(n)} \text{ s.t. } \mathsf{cost}(e) > \mathsf{OPT}/n\}$ are pinned edges with non-negligible costs. Note that A will only grow at Lines 7 and 12 in Algorithm 1.

Line 7. Consider an execution of Line 7 that brings new A'-pinned edges. The current non-inherited original edge set $E_{\mathrm{orig}}(\hat{e}^{(t)})$ must come from the top $\lceil \log n \rceil + 2$ levels (i.e., the current level i is between $L^{(n)} - \lceil \log n \rceil - 1$ and the maximum level $L^{(n)}$). Otherwise, we have $\mathsf{cost}(E_{\mathrm{orig}}(\hat{e}^{(t)})) \le 2^{i+1} \le 2^{L^{(n)}-1}/n \le \mathsf{OPT}/n$ (note that $2^{L^{(n)}-1} \le \mathsf{OPT}$ by the definition of the function level), and any pinned edge from $E_{\mathrm{orig}}(\hat{e}^{(t)})$ would have negligible cost. Furthermore, the algorithm guarantees that the total cost of new A'-pinned edges from this execution is at most $\mathsf{cost}(E_{\mathrm{orig}}(\hat{e}^{(t)}))/\lambda$.

Line 12. Suppose an execution of Line 12 brings a new A'-pinned edge. Note that at this moment, the buffer B is the *multiset union* of several non-inherited original edge sets, all of which come from the top $\lceil \log n \rceil + 2$ levels by the same argument as in the previous case. Again, the algorithm guarantees the cost of this new A'-pinned edge is at most $\mathsf{cost}(B)/\lambda$.

Finally, observe that the sum of $\mathsf{cost}(E_{\mathrm{orig}}(\hat{e}^{(t)}))$ over such executions of Line 7 (which brings new A' edges) and $\mathsf{cost}(B)$ over such executions of Line 12 is at most $O(\log n \cdot \mathsf{OPT})$ by Theorem 14 (since the involved non-inherited original edge sets are all from the top $\lceil \log n \rceil + 2$ levels), so we get $\mathsf{cost}(A') \le O(\log n \cdot \mathsf{OPT}/\lambda)$ and $\mathsf{cost}(A) \le \mathsf{cost}(A') + O(\mathsf{OPT}) \le O(\log n \cdot \mathsf{OPT}/\lambda)$.

4.6 Proof of Theorem 14

We will invoke a fact that forms the initial part of the analysis in [55]. By losing a constant factor in cost, it allows us to assume that the clustering induced by our solution is a refinement of that of the optimal solution (in [55] this is called the faithfulness property). For clarity, we refer to $\mathscr{C}^{(n)}_{L^{(n)}+1}$ as $\mathscr{C}^{(n)}_{\max}$, which is the top-level clustering of the hierarchy $\mathscr{H}^{(n)}$ after the last arrival.

▶ **Lemma 15** (Theorem 4.1 in [55]). *There exists a solution $F^{\star\star}$ of the instance $(\mathcal{M}, D)$ satisfying that*

- *the cost of $F^{\star\star}$ is at most $O(\mathsf{OPT})$, and*
- *for each cluster $C \in \mathscr{C}^{(n)}_{\max}$, all vertices in C belong to the same connected component of $F^{\star\star}$.*

Now we prove Theorem 14 using Lemma 15. We restate Theorem 14 for ease of reading.

▶ **Theorem 14.** *For each level i we have*

$$\sum_t |\hat{F}_i^{(t)} \setminus \hat{F}_{\mathrm{inh},i}^{(t)}| \cdot 2^{i+1} \leq O(\mathsf{OPT})\,. \tag{3}$$

Proof. We use a dual fitting argument.

Primal LP and its Dual. Consider the following primal LP, where $\mathcal{S}$ is the family of all cuts $S \subseteq V$ separating at least one cluster in $\mathscr{C}_{\max}^{(n)}$, i.e., there exists a $C \in \mathscr{C}_{\max}^{(n)}$ with $C \cap S \neq \emptyset$ and $C \setminus S \neq \emptyset$. For each cut $S \in \mathcal{S}$, we use $\partial(S) \subseteq E$ to denote the set of original edges crossing S.

$$\begin{aligned} \min \quad & \sum_{e \in E} \mathsf{cost}(e) \cdot x(e) \\ \text{s.t.} \quad & \sum_{e \in \partial(S)} x(e) \geq 1 \qquad \forall S \in \mathcal{S} \\ & x \geq 0. \end{aligned}$$

Note that this primal LP does not exactly correspond to the instance $(\mathcal{M}, D)$, since it requires the solution to have stronger connectivity (each cluster in $\mathscr{C}_{\max}^{(n)}$ should have all its vertices connected). However, its optimal value $\mathsf{OPT}_{\mathrm{LP}}$ will not exceed OPT too much, i.e., we have

$$\mathsf{OPT}_{\mathrm{LP}} \leq O(\mathsf{OPT}),$$

because the solution $F^{\star\star}$ in Lemma 15 gives a feasible solution (for each $e \in E$, set $x(e) = 1$ if $e \in F^{\star\star}$, otherwise $x(e) = 0$) to the LP with cost at most $O(\mathsf{OPT})$. The dual LP is as follows:

$$\begin{aligned} \max \quad & \sum_{S \in \mathcal{S}} y(S) \\ \text{s.t.} \quad & \sum_{S\,:\,e \in \partial(S)} y(S) \leq \mathsf{cost}(e) \qquad \forall e \in E \\ & y \geq 0. \end{aligned}$$

An intermediate goal is to construct a feasible dual solution with value approximately the LHS of (3). We slightly abuse notation by viewing a dual solution as a function $y : 2^V \to \mathbb{R}_{\geq 0}$, and we say such a dual solution y is feasible if it satisfies the dual constraints and is zero outside the actual domain $\mathcal{S}$.

Construction of the Dual Solution y. We start with some standard terminology of the dual-fitting argument. Consider a vertex $v \in V$, and let $u_1, u_2, ..., u_{2n}$ be an order of vertices with $\mathsf{dist}_{\mathcal{M}}(v, u_j)$ increasing (in particular $u_1 = v$). By *growing a ball with radius r around v*, we mean for each u_j with $\mathsf{dist}_{\mathcal{M}}(v, u_j) \leq r$, adding a value $\min\{\mathsf{dist}_{\mathcal{M}}(v, u_{j+1}), r\} - \mathsf{dist}_{\mathcal{M}}(v, u_j)$ to the dual variable $y(\{u_1, ..., u_j\})$. The following observation is straightforward by the triangle inequality.

▶ **Observation 16.** *Let $X \subseteq V$ be a set of vertices with pairwise distances at least $2r$ in $\mathcal{M}$. Then the dual solution obtained by growing a ball of radius r around each vertex in X satisfies all the dual constraints.*

The dual solution we will construct is exactly by growing a ball of radius r around each vertex in a subset $X \subseteq V$. We set the radius

$$r = 2^{i-1},$$

and define the subset $X \subseteq V$ by the following procedure.

The procedure will define subsets $\hat{X}^{(t)}$ and $X^{(t)}$ for each arrival t. We will take the final X to be $X^{(n)}$ (corresponding to the last arrival). The subsets will be defined so as to satisfy the following invariants:

1. $\hat{X}^{(t)}$ is contained in the union of i-active clusters in $\mathscr{C}_{i+1}^{(t)}$. Recall that i-active clusters are the clusters C with $\mathsf{level}(C) \geq i$.
2. Vertices in $\hat{X}^{(t)}$ have pairwise distances at least $2r$ in $\mathcal{M}$.
3. $X^{(t)} \subseteq \hat{X}^{(t)}$, and for each i-active cluster $C \in \mathscr{C}_{i+1}^{(t)}$, $\hat{X}^{(t)} \cap C$ has at least one vertex outside $X^{(t)}$.
4. $|X^{(t)}| = \sum_{t' \leq t} |\hat{F}_i^{(t')} \setminus \hat{F}_{\mathrm{inh},i}^{(t')}|$.

In particular, if some arrival t has maximum level $L^{(t)}$ less than i, then $\hat{X}^{(t)}$ and $X^{(t)}$ are simply empty, which clearly satisfy all the invariants. In what follows, we first describe the construction of $\hat{X}^{(t)}$ and $X^{(t)}$, and then argue why taking $X := X^{(n)}$ gives a feasible dual solution.

We now construct $\hat{X}^{(t)}$ and $X^{(t)}$ assuming that $\hat{X}^{(t-1)}$ and $X^{(t-1)}$ (satisfying the invariants) are given. Recall the following from Section 4.2: $\mathscr{C}_{i+1}^{(t)}$ is exactly the clustering obtained by contracting $\hat{F}_i^{(t)}$ over $\mathscr{C}_i^{(t)}$. The virtual edges $\hat{F}_{\mathrm{inh},i}^{(t)} \subseteq \hat{F}_i^{(t)}$ are inherited. $\mathscr{C}_{\mathrm{inh},i}^{(t)}$ is an intermediate clustering obtained by contracting $\hat{F}_{\mathrm{inh},i}^{(t)}$ over $\mathscr{C}_i^{(t)}$.

Useful Properties on Clusterings. The argument below will heavily exploit some useful properties of clusterings $\mathscr{C}_i^{(t-1)}, \mathscr{C}_{i+1}^{(t-1)}, \mathscr{C}_i^{(t)}, \mathscr{C}_{\mathrm{inh},i}^{(t)}$ and $\mathscr{C}_{i+1}^{(t)}$. First, directly from the clustering procedure, we have

$$\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_{i+1}^{(t-1)}, \qquad \text{and} \qquad \mathscr{C}_i^{(t)} \preceq \mathscr{C}_{\mathrm{inh},i}^{(t)} \preceq \mathscr{C}_{i+1}^{(t)}.$$

Second, recall that Lemma 9 and Lemma 11 provide

$$\mathscr{C}_i^{(t-1)} \preceq \mathscr{C}_i^{(t)}, \qquad \text{and} \qquad \mathscr{C}_{i+1}^{(t-1)} \preceq \mathscr{C}_{\mathrm{inh},i}^{(t)}.$$

We sometimes consider the union of i-active clusters in these clusterings. Let $\mathsf{ActUn}(\mathscr{C}')$ denote the union of i-active clusters in a clustering $\mathscr{C}'$. We can easily observe that

$$\mathsf{ActUn}(\mathscr{C}_i^{(t-1)}) = \mathsf{ActUn}(\mathscr{C}_{i+1}^{(t-1)}) \subseteq \mathsf{ActUn}(\mathscr{C}_i^{(t)}) = \mathsf{ActUn}(\mathscr{C}_{\mathrm{inh},i}^{(t)}) = \mathsf{ActUn}(\mathscr{C}_{i+1}^{(t)})$$

by the clustering procedure and the above properties.

Construction of $\hat{X}^{(t)}$. Let $(u^{(t)}, v^{(t)})$ be the new demand pair of arrival t. Initially, set $\hat{X}^{(t)} = \hat{X}^{(t-1)}$. If $\mathsf{level}(u^{(t)}), \mathsf{level}(v^{(t)}) \geq i$, let $C_u, C_v \in \mathscr{C}_{\mathrm{inh},i}^{(t)}$ be the clusters containing $u^{(t)}$ and $v^{(t)}$ respectively (possibly C_u and C_v are the same cluster).

- If C_u and C_v are the same cluster, add $u^{(t)}$ into $\hat{X}^{(t)}$ if $u^{(t)}$ and $v^{(t)}$ are the only vertices with level at least i in C_u.
- If C_u and C_v are different clusters, add $u^{(t)}$ (resp. $v^{(t)}$) into $\hat{X}^{(t)}$ if $u^{(t)}$ (resp. $v^{(t)}$) are the only vertices with level at least i in C_u (resp. C_v).

▶ **Lemma 17.** *We have the following.*

1. *$\hat{X}^{(t)}$ is contained in the union of i-active clusters in $\mathscr{C}_{\mathrm{inh},i}^{(t)}$, i.e., $\hat{X}^{(t)} \subseteq \mathsf{ActUn}(\mathscr{C}_{\mathrm{inh},i}^{(t)})$.*
2. *Vertices in $\hat{X}^{(t)}$ have pairwise distances at least $2r$ in $\mathcal{M}$.*
3. *For each i-active cluster $C \in \mathscr{C}_{\mathrm{inh},i}^{(t)}$, $\hat{X}^{(t)} \cap C$ has at least one vertex outside $X^{(t-1)}$.*

Proof. Item 1 is straightforward from the construction of $\hat{X}^{(t)}$: the new vertices of $\hat{X}^{(t)}$ (i.e., one or more of $u^{(t)}$ and $v^{(t)}$) always come from the i-active clusters $C_u, C_v \in \mathscr{C}^{(t)}_{\mathrm{inh},i}$; the old vertices of $\hat{X}^{(t)}$ satisfy $\hat{X}^{(t-1)} \subseteq \mathsf{ActUn}(\mathscr{C}^{(t-1)}_{i+1})$ (by Invariant 1), and thus $\hat{X}^{(t-1)} \subseteq \mathsf{ActUn}(\mathscr{C}^{(t)}_{\mathrm{inh},i})$.

Next, we show Item 2. Suppose C_u and C_v are different clusters (an analogous argument works for the case $C_u = C_v$). It suffices to show that if we add $u^{(t)}$ into $\hat{X}^{(t)}$, this new $u^{(t)}$ is far from any vertices in $\hat{X}^{(t-1)}$. Recall that we add $u^{(t)}$ only if $u^{(t)}$ is the only vertex with level at least i in C_u. This means the i-active cluster $C'_u \in \mathscr{C}^{(t)}_i$ containing $u^{(t)}$ also has $u^{(t)}$ as the only vertex with level at least i (since $C'_u \subseteq C_u$), and thus, C'_u is disjoint from any i-active cluster in $\mathscr{C}^{(t-1)}_i$ (since $\mathscr{C}^{(t-1)}_i \preceq \mathscr{C}^{(t)}_i$). Therefore, C'_u is disjoint from $\mathsf{ActUn}(\mathscr{C}^{(t-1)}_i) = \mathsf{ActUn}(\mathscr{C}^{(t-1)}_{i+1}) \supseteq \hat{X}^{(t-1)}$.

Now consider any vertex $x \in \hat{X}^{(t-1)}$. It must belong to an i-active $\mathscr{C}^{(t)}_i$-cluster C'_x (since $\hat{X}^{(t-1)} \subseteq \mathsf{ActUn}(\mathscr{C}^{(t-1)}_{i+1}) \subseteq \mathsf{ActUn}(\mathscr{C}^{(t)}_i)$), and this C'_x cannot be the same as C'_u (since C'_u is disjoint from $\hat{X}^{(t-1)}$). Hence, we can conclude

$$\mathsf{dist}_{\mathcal{M}}(x, u^{(t)}) \geq \mathsf{dist}_{\mathcal{M}/\mathscr{C}^{(t)}_i}(C'_x, C'_u) \geq 2^i$$

as desired, where the second inequality is by Observation 5.

Regarding Item 3, if an i-active cluster $C \in \mathscr{C}^{(t)}_{\mathrm{inh},i}$ contains an i-active cluster in $\mathscr{C}^{(t-1)}_{i+1}$, then by Invariant 3 of $\hat{X}^{(t-1)}$, $\hat{X}^{(t-1)} \cap C$ will have at least one vertex outside $X^{(t-1)}$. Thus, we only need to pay attention to those i-active clusters $C \in \mathscr{C}^{(t)}_{\mathrm{inh},i}$ containing no i-active cluster in $\mathscr{C}^{(t-1)}_{i+1}$. Because $\mathscr{C}^{(t-1)}_{i+1} \preceq \mathscr{C}^{(t)}_{\mathrm{inh},i}$ (Lemma 11), such a cluster C must have no vertex with level at least i from $V^{(t-1)}$. In other words, such a cluster C exists only when $\mathsf{level}(u^{(t)}) = \mathsf{level}(v^{(t)}) \geq i$ (since C is i-active) and it must contain either $u^{(t)}$ or $v^{(t)}$ (or both). Then by our construction of $\hat{X}^{(t)}$, we indeed add a new vertex from C into $\hat{X}^{(t)}$ as desired. ◀

Construction of $X^{(t)}$. Initially, set $X^{(t)} = X^{(t-1)}$.

▶ **Observation 18.** *Each i-active cluster in $\mathscr{C}^{(t)}_{\mathrm{inh},i}$ is contained by an i-active cluster in $\mathscr{C}^{(t)}_{i+1}$. Each i-active cluster in $\mathscr{C}^{(t)}_{i+1}$ only contains i-active clusters in $\mathscr{C}^{(t)}_{\mathrm{inh},i}$.*

Proof. This is simply because, from $\mathscr{C}^{(t)}_{\mathrm{inh},i}$ to $\mathscr{C}^{(t)}_{i+1}$, the clustering procedure only merges i-active clusters. ◀

For each i-active cluster $C \in \mathscr{C}^{(t)}_{i+1}$, if it is made up of k many i-active clusters in $\mathscr{C}^{(t)}_{\mathrm{inh},i}$, then by Item 3 in Lemma 17, $\hat{X}^{(t)} \cap C$ has at least k vertices outside $X^{(t-1)}$, and we add arbitrary $k-1$ of them into $X^{(t)}$.

Proving Invariants of $\hat{X}^{(t)}$ and $X^{(t)}$. Invariant 1 is by Item 1 in Lemma 17 and the fact that the union of i-active clusters in $\mathscr{C}^{(t)}_{\mathrm{inh},i}$ is the same as that in $\mathscr{C}^{(t)}_{i+1}$. Invariant 2 has also been proven in Lemma 17. Invariant 3 is by the construction of $X^{(t)}$.

Finally, to see Invariant 4, it suffices to show that $|X^{(t)}| - |X^{(t-1)}| = |\hat{F}^{(t)}_i| - |\hat{F}^{(t)}_{\mathrm{inh},i}|$. Furthermore, we can observe that $|\hat{F}^{(t)}_i| = |\mathscr{C}^{(t)}_i| - |\mathscr{C}^{(t)}_{i+1}|$, since $\mathscr{C}^{(t)}_{i+1}$ is obtained by contracting $\hat{F}^{(t)}_i$ over $\mathscr{C}^{(t)}_i$, and similarly, we have $|\hat{F}^{(t)}_{\mathrm{inh},i}| = |\mathscr{C}^{(t)}_i| - |\mathscr{C}^{(t)}_{\mathrm{inh},i}|$. By the construction of $X^{(t)}$, the number of new vertices added to $X^{(t)}$, i.e., $|X^{(t)}| - |X^{(t-1)}|$, is exactly the number of i-active clusters in $\mathscr{C}^{(t)}_{\mathrm{inh},i}$ minus the number of i-active clusters in $\mathscr{C}^{(t)}_{i+1}$, which is the same as $|\mathscr{C}^{(t)}_{\mathrm{inh},i}| - |\mathscr{C}^{(t)}_{i+1}|$ since the i-inactive clusters in $\mathscr{C}^{(t)}_{\mathrm{inh},i}$ are the same as those in $\mathscr{C}^{(t)}_{i+1}$. Combining all the above, we get $|X^{(t)}| - |X^{(t-1)}| = |\hat{F}^{(t)}_i| - |\hat{F}^{(t)}_{\mathrm{inh},i}|$.

Feasibility of the Dual Solution y. Recall that we have two steps to show the feasibility of the dual solution y. First, it is required that y satisfies all the dual constraints, which is indeed the case by Observation 16 and the way we construct y. Second, we also need to show that y is zero outside the actual domain $\mathcal{S}$ (recall that $\mathcal{S}$ collects all cuts separating at least one cluster in $\mathscr{C}^{(n)}_{\max}$).

Consider a vertex $v \in X$. When we grow a ball with radius r around v, the cuts S for which we may add a positive value to $y(S)$ must satisfy $S \subseteq \mathsf{Ball}(v, r) = \{u \in V^{(n)} \mid \mathsf{dist}_{\mathcal{M}}(v, u) \le r\}$. However, by Invariants 1, 2 and 3, in the cluster $C_v \in \mathscr{C}^{(n)}_{i+1}$ containing v, there must be another vertex $v' \in C_v \cap (\hat{X}^{(n)} \setminus X)$ such that $\mathsf{dist}_{\mathcal{M}}(v, v') \ge 2r$. Hence v' is outside all such cuts S, which means that all such S will separate C_v. Finally, since $\mathscr{C}^{(n)}_{i+1} \preceq \mathscr{C}^{(n)}_{\max}$, all such S will separate the $\mathscr{C}^{(n)}_{\max}$-cluster containing C_v, as desired.

Completing the Proof. By Invariant 4, the value of the dual solution y is

$$D = \sum_S y(S) = |X| r = \sum_t |\hat{F}^{(t)}_i \setminus \hat{F}^{(t)}_{\mathrm{inh},i}| \cdot 2^{i-1}.$$

Combining it with $D \le \mathsf{OPT}_{\mathrm{LP}} \le O(\mathsf{OPT})$, we get the original lemma. ◀

5 Conclusion

In this work we initiate the study of low-recourse algorithms for online Steiner forest. We gave a constant-competitive algorithm with $O(\log n)$ amortized recourse. This prompts several natural follow-up questions:

▶ **Question 19.** *Is there an algorithm with $O(1)$ recourse?*

▶ **Question 20.** *Can the recourse be de-amortized?*

▶ **Question 21.** *Can one handle the fully-dynamic setting, where terminal pairs both arrive and depart? (What about the deletion-only model?)*

For completeness, we also remark that for the Steiner *tree* problem, an algorithm with worst-case (un-amortized) constant recourse is not known for the fully dynamic setting.

▶ Remark 22. Obtaining $O(\log n)$ recourse in the *deletion-only setting* (of online Steiner forest) is straightforward, at least if we assume that the optimal cost is bounded polynomially in n: one can recompute the solution whenever the optimal cost decreases by a factor 2.[6]

References

1 Amir Abboud, Raghavendra Addanki, Fabrizio Grandoni, Debmalya Panigrahi, and Barna Saha. Dynamic set cover: improved algorithms and lower bounds. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 114–125. ACM, 2019. doi:10.1145/3313276.3316376.

[6] We can further remove the assumption by *bucketing* the demand pairs. That is, initially, we put a pair (u, v) with $\mathsf{dist}_{\mathcal{M}}(u, v) \in [n^i, n^{i+1})$ into the i-th bucket, and then for each bucket i, create its own initial solution F_i (let the entire initial solution be the union of all F_i). Note that $|F_i|$ should be proportional to the bucket size, since the original graph is complete with edge weights forming a metric. With this initial solution, whenever the optimal cost is halved, we only need to recompute the F_i of the top $O(1)$ buckets. Each F_i is recomputed $O(\log n)$ times, leading to $O(\log n)$ amortized recourse.

2 Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. doi:10.1137/S0097539792236237.

3 Ali Ahmadi, Iman Gholami, MohammadTaghi Hajiaghayi, Peyman Jabbarzade, and Mohammad Mahdavi. Breaking a long-standing barrier: 2-ε approximation for steiner forest. In *66th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2025, Sydney, Australia, December 14-17, 2025*, pages 373–444. IEEE, 2025. doi:10.1109/FOCS63196.2025.00023.

4 Matthew Andrews, Michel X. Goemans, and Lisa Zhang. Improved bounds for on-line load balancing. *Algorithmica*, 23(4):278–301, 1999. doi:10.1007/PL00009263.

5 Spyros Angelopoulos, Christoph Dürr, and Shendan Jin. Online maximum matching with recourse. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 8:1–8:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.8.

6 Sepehr Assadi, Krzysztof Onak, Baruch Schieber, and Shay Solomon. Fully dynamic maximal independent set with sublinear update time. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 815–826. ACM, 2018. doi:10.1145/3188745.3188922.

7 Sepehr Assadi and Shay Solomon. Fully dynamic set cover via hypergraph maximal matching: An optimal approximation through a local approach. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.8.

8 Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. In Éva Tardos, editor, *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 28-30 January 1996, Atlanta, Georgia, USA*, pages 68–74. ACM/SIAM, 1996. URL: http://dl.acm.org/citation.cfm?id=313852.313888.

9 Baruch Awerbuch, Yossi Azar, Serge A. Plotkin, and Orli Waarts. Competitive routing of virtual circuits with unknown duration. *J. Comput. Syst. Sci.*, 62(3):385–397, 2001. doi:10.1006/jcss.1999.1662.

10 Étienne Bamas, Marina Drygala, and Andreas Maggiori. An improved analysis of greedy for online steiner forest. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 3202–3229. SIAM, SIAM, 2022. doi:10.1137/1.9781611977073.125.

11 Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. Fully dynamic randomized algorithms for graph spanners. *ACM Trans. Algorithms*, 8(4), October 2012. doi:10.1145/2344422.2344425.

12 Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Cliff Stein, and Madhu Sudan. Fully dynamic maximal independent set with polylogarithmic update time. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 382–405. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00032.

13 Suman K. Bera, Sayan Bhattacharya, Jayesh Choudhari, and Prantar Ghosh. A new dynamic algorithm for densest subhypergraphs. In Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini, editors, *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, pages 1093–1103. ACM, 2022. doi:10.1145/3485447.3512158.

14 Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 344–353, New York, NY, USA, 1997. Association for Computing Machinery. doi:10.1145/258533.258618.

15 Aaron Bernstein, Jacob Holm, and Eva Rotenberg. Online bipartite matching with amortized o(log 2 n) replacements. *J. ACM*, 66(5), September 2019. doi:10.1145/3344999.

16 Aaron Bernstein, Tsvi Kopelowitz, Seth Pettie, Ely Porat, and Clifford Stein. Simultaneously Load Balancing for Every p-norm, With Reassignments. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2017.51.

17 Sayan Bhattacharya, Niv Buchbinder, Roie Levin, and Thatchaphol Saranurak. Chasing positive bodies. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1694–1714, 2023. doi:10.1109/FOCS57990.2023.00103.

18 Sayan Bhattacharya, Ruoxu Cen, and Debmalya Panigrahi. Fully dynamic set cover: Worst-case recourse and update time. *CoRR*, abs/2511.08485, 2025. To appear in Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC 2026). doi:10.48550/arXiv.2511.08485.

19 Sayan Bhattacharya, Fabrizio Grandoni, and David Wajc. Online edge coloring algorithms via the nibble method. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA '21)*, pages 2830–2841. SIAM, 2021. doi:10.1137/1.9781611976465.168.

20 Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. A new deterministic algorithm for dynamic set cover. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 406–423. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00033.

21 Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Xiaowei Wu. Dynamic set cover: Improved amortized and worst-case update time. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2537–2549. SIAM, 2021. doi:10.1137/1.9781611976465.150.

22 Sayan Bhattacharya, Silvio Lattanzi, and Nikos Parotsidis. Efficient and stable fully dynamic facility location. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/943d6dca1884955e645d8997ae2fa938-Abstract-Conference.html.

23 Sayan Bhattacharya, Thatchaphol Saranurak, and Pattara Sukprasert. Simple dynamic spanners with near-optimal recourse against an adaptive adversary. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 17:1–17:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.17.

24 Sujoy Bhore, Arnold Filtser, and Csaba D. Tóth. Online duet between metric embeddings and minimum-weight perfect matchings. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4564–4579. SIAM, 2024. doi:10.1137/1.9781611977912.162.

25 Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 384–393. IEEE, 2014. doi:10.1109/FOCS.2014.48.

26 Gerth Stølting Brodal and Rolf Fagerberg. Dynamic representations of sparse graphs. In Frank Dehne, Jörg-Rüdiger Sack, Arvind Gupta, and Roberto Tamassia, editors, *Algorithms and Data Structures*, pages 342–351, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. doi:10.1007/3-540-48447-7_34.

27 Niv Buchbinder, Anupam Gupta, Daniel Hathcock, Anna R. Karlin, and Sherry Sarkar. Maintaining matroid intersections online. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 4283–4304. SIAM, 2024. doi:10.1137/1.9781611977912.149.

28 Niv Buchbinder, Roie Levin, and Yue Yang. Competitively consistent clustering. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL: https://proceedings.mlr.press/v267/buchbinder25a.html.

29 Niv Buchbinder, Joseph Naor, and David Wajc. Chasing submodular objectives, and submodular maximization via cutting planes. *CoRR*, abs/2511.13605, 2025. doi:10.48550/arXiv.2511.13605.

30 Anton Bukov, Shay Solomon, and Tianyi Zhang. Nearly optimal dynamic set cover: Breaking the quadratic-in-f time barrier. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 824–863. SIAM, 2025. doi:10.1137/1.9781611978322.24.

31 Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanita. An improved lp-based approximation for steiner tree. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 583–592, 2010. doi:10.1145/1806689.1806769.

32 Hans-Joachim Böckenhauer, Ralf Klasing, Tobias Mömke, Peter Rossmanith, Moritz Stocker, and David Wehner. Online knapsack with removal and recourse. *Journal of Computer and System Sciences*, 155:103697, 2026. doi:10.1016/j.jcss.2025.103697.

33 Keren Censor-Hillel, Elad Haramaty, and Zohar S. Karnin. Optimal dynamic distributed MIS. In George Giakkoupis, editor, *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 217–226. ACM, 2016. doi:10.1145/2933057.2933083.

34 T-H. Hubert Chan, Shaofeng H.-C. Jiang, Tianyi Wu, and Mengshi Zhao. Online clustering with nearly optimal consistency. In *The Thirteenth International Conference on Learning Representations*, 2025. URL: https://openreview.net/forum?id=NA2vUMaMOm.

35 Kamalika Chaudhuri, Constantinos Daskalakis, Robert D. Kleinberg, and Henry Lin. Online bipartite perfect matching with augmentations. In *IEEE INFOCOM 2009*, pages 1044–1052. IEEE, 2009. doi:10.1109/INFCOM.2009.5062016.

36 Ho-Lin Chen, Tim Roughgarden, and Gregory Valiant. Designing network protocols for good equilibria. *SIAM Journal on Computing*, 39(5):1799–1832, 2010. doi:10.1137/08072721X.

37 Miroslav Chlebík and Janka Chlebíková. The steiner tree problem on graphs: Inapproximability results. *Theoretical Computer Science*, 406(3):207–214, 2008. doi:10.1016/j.tcs.2008.06.046.

38 Vincent Cohen-Addad, Niklas Oskar D. Hjuler, Nikos Parotsidis, David Saulpic, and Chris Schwiegelshohn. Fully dynamic consistent facility location. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3250–3260, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/fface8385abbf94b4593a0ed53a0c70f-Abstract.html.

39 Mark de Berg, Arpan Sadhukhan, and Frits Spieksma. Stable approximation algorithms for the dynamic broadcast range-assignment problem. *SIAM Journal on Discrete Mathematics*, 38(1):790–827, 2024. doi:10.1137/23M1545975.

40 Paul Duetting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, and Morteza Zadimoghaddam. Consistent submodular maximization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=AlJkqMnyjL.

41 Paul Dütting, Federico Fusco, Silvio Lattanzi, Ashkan Norouzi-Fard, Ola Svensson, and Morteza Zadimoghaddam. The cost of consistency: Submodular maximization with constant recourse. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 1406–1417. ACM, 2025. doi:10.1145/3717823.3718131.

42 Leah Epstein and Asaf Levin. Robust algorithms for preemptive scheduling. *Algorithmica*, 69(1):26–57, 2014. doi:10.1007/s00453-012-9718-3.

43 Hendrik Fichtenberger, Silvio Lattanzi, Ashkan Norouzi-Fard, and Ola Svensson. Consistent k-clustering for general metrics. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA '21)*, pages 2660–2678. SIAM, 2021. doi:10.1137/1.9781611976465.158.

44 Sebastian Forster and Antonis Skarlatos. Dynamic consistent k-center clustering with optimal recourse. In *Proceedings of the 2025 ACM-SIAM Symposium on Discrete Algorithms (SODA '25)*, pages 212–254. SIAM, 2025. doi:10.1137/1.9781611978322.7.

45 Ayoub Foussoul, Vineet Goyal, and Amit Kumar. Fully-dynamic load balancing. In Jens Vygen and Jaroslaw Byrka, editors, *Integer Programming and Combinatorial Optimization - 25th International Conference, IPCO 2024, Wroclaw, Poland, July 3-5, 2024, Proceedings*, volume 14679 of *Lecture Notes in Computer Science*, pages 182–195. Springer, 2024. doi:10.1007/978-3-031-59835-7_14.

46 Waldo Gálvez, José A. Soto, and José Verschae. Symmetry exploitation for online machine covering with bounded migration. *ACM Trans. Algorithms*, 16(4):43:1–43:22, July 2020. doi:10.1145/3397535.

47 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

48 Martin Groß, Anupam Gupta, Amit Kumar, Jannik Matuschke, Daniel R. Schmidt, Melanie Schmidt, and José Verschae. A local-search algorithm for steiner forest. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 31:1–31:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.31.

49 Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. Online perfect matching and mobile computing. In *Workshop on Algorithms and Data Structures*, pages 194–205. Springer, 1995. doi:10.1007/3-540-60220-8_62.

50 Albert Gu, Anupam Gupta, and Amit Kumar. The power of deferral: maintaining a constant-competitive steiner tree online. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 525–534. ACM, 2013. doi:10.1145/2488608.2488674.

51 Xiangyu Guo, Janardhan Kulkarni, Shi Li, and Jiayi Xian. On the facility location problem in online and dynamic models. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 42:1–42:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.42.

52 Anupam Gupta, Vijaykrishna Gurunathan, Ravishankar Krishnaswamy, Amit Kumar, and Sahil Singla. Online discrepancy with recourse for vectors and graphs. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 1356–1383. SIAM, 2022. doi:10.1137/1.9781611977073.57.

53 Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 537–550, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3055399.3055493.

54 Anupam Gupta and Amit Kumar. Online steiner tree with deletions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 455–467. SIAM, 2014. doi:10.1137/1.9781611973402.34.

55 Anupam Gupta and Amit Kumar. Greedy algorithms for steiner forest. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 871–878. ACM, 2015. doi:10.1145/2746539.2746590.

56 Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining assignments online: Matching, scheduling, and flows. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 468–479. SIAM, 2014. doi:10.1137/1.9781611973402.35.

57 Anupam Gupta and Roie Levin. Fully-dynamic submodular cover with bounded recourse. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1147–1157. IEEE, 2020. doi:10.1109/FOCS46700.2020.00110.

58 Anupam Gupta and Vera Traub. Steiner forest: A simplified better-than-2 approximation. *CoRR*, abs/2511.18460, 2025. To appear in Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC 2026). doi:10.48550/arXiv.2511.18460.

59 Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM J. Discret. Math.*, 4(3):369–384, 1991. doi:10.1137/0404033.

60 Mohammad Reza Karimi Jaghargh, Andreas Krause, Silvio Lattanzi, and Sergei Vassilvtiskii. Consistent online optimization: Convex and submodular. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2241–2250. PMLR, 16–18 April 2019. URL: https://proceedings.mlr.press/v89/jaghargh19a.html.

61 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of Symposium on the Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

62 Lawrence T. Kou, George Markowsky, and Leonard Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15:141–145, 1981. doi:10.1007/BF00288961.

63 Ravishankar Krishnaswamy, Shi Li, and Varun Suriyanarayana. Online unrelated-machine load balancing and generalized flow with recourse. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 775–788. ACM, 2023. doi:10.1145/3564246.3585222.

64 Jakub Łącki, Bernhard Haeupler, Christoph Grunau, Rajesh Jayaram, and Václav Rozhoň. Fully dynamic consistent k-center clustering. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3463–3484. SIAM, 2024. doi:10.1137/1.9781611977912.124.

65 Jakub Łącki, Jakub Oćwieja, Marcin Pilipczuk, Piotr Sankowski, and Anna Zych. The power of dynamic distance oracles: Efficient dynamic algorithms for the steiner tree. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 11–20, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2746539.2746615.

66 Silvio Lattanzi and Sergei Vassilvitskii. Consistent k-clustering. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1975–1984. PMLR, August 2017. URL: https://proceedings.mlr.press/v70/lattanzi17a.html.

67 Nicole Megow and Lukas Nölke. Online Minimum Cost Matching with Recourse on the Line. In Jarosław Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, volume 176 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.37.

68 Nicole Megow, Martin Skutella, José Verschae, and Andreas Wiese. The power of recourse for online MST and TSP. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 689–700. Springer, 2012. doi:10.1007/978-3-642-31594-7_58.

69 Steven Phillips and Jeffery Westbrook. Online load balancing and network flow. *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*, pages 402–411, 1993. doi:10.1145/167088.167201.

70 Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009. doi:10.1287/moor.1090.0381.

71 Saurabh Sawlani and Junxing Wang. Near-optimal fully dynamic densest subgraph. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 181–193. ACM, 2020. doi:10.1145/3357713.3384327.

72 Martin Skutella and José Verschae. A robust PTAS for machine covering and packing. In Mark de Berg and Ulrich Meyer, editors, *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, volume 6346 of *Lecture Notes in Computer Science*, pages 36–47. Springer, 2010. doi:10.1007/978-3-642-15775-2_4.

73 Noam Solomon and Shay Solomon. A generalized matching reconfiguration problem. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 57:1–57:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.57.

74 Shay Solomon and Amitai Uzrad. Dynamic ((1+ϵ) ln n)-approximation algorithms for minimum set cover and dominating set. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1187–1200. ACM, 2023. doi:10.1145/3564246.3585211.

75 Shay Solomon and Amitai Uzrad. Dynamic set cover with worst-case recourse. *CoRR*, abs/2511.07354, 2025. doi:10.48550/arXiv.2511.07354.

76 Shay Solomon, Amitai Uzrad, and Tianyi Zhang. A lossless deamortization for dynamic greedy set cover. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 264–290. IEEE, 2024. doi:10.1109/FOCS61266.2024.00025.

77 Shay Solomon and Nicole Wein. Improved dynamic graph coloring. *ACM Trans. Algorithms*, 16(3), June 2020. doi:10.1145/3392724.

78 Jeffery Westbrook. Load balancing for response time. *Journal of Algorithms*, 35(1):1–16, 2000. doi:10.1006/jagm.2000.1074.

79 Jeffery R. Westbrook and Dicky C. K. Yan. The performance of greedy algorithms for the on-line steiner tree and related problems. *Mathematical Systems Theory*, 28(5):451–468, 1995. doi:10.1007/BF01185867.

A The Online (Timed) Gluttonous Algorithm

In this section, we give the formal description of an online simulation of the timed gluttonous algorithm in [55], which is a classic offline Steiner forest algorithm. We can show that it is $O(\log n)$-competitive.

Algorithm 3 Online (Timed) Gluttonous.

Define $\mathsf{level}(\cdot)$ for terminals and clusters as in Section 3.
Initialize $\mathscr{C}$ to be an empty clustering of the initially empty terminal set.
while a demand pair (u_t, v_t) arrives **do**
    Add two clusters $\{u_t\}, \{v_t\}$ into $\mathscr{C}$
    for $i = 0$ to the max level **do**
        while exist i-active clusters $C_1, C_2 \in \mathscr{C}$ s.t. $\mathsf{dist}_{\mathcal{M}/\mathscr{C}}(C_1, C_2) < 2^{i+1}$ **do**
            In $\mathscr{C}$, replace C_1 and C_2 with $C_1 \cup C_2$
        end while
    end for
end while

We can use the same idea as in the proof of Theorem 14 to prove an analogue of Theorem 14: for each level i, the number of level-i merges times 2^{i+1} is at most $O(\mathsf{OPT})$. Therefore, this algorithm is $O(\log n)$-competitive since the total cost of merges from outside of the top $O(\log n)$ levels is negligible (since each level has at most $O(n)$ merges, and the cost per merge decreases exponentially as i goes down).