To Buy or Not to Buy: Online Rent-Or-Buy on Node-Weighted Graphs

Sander Borst
Max-Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Moritz Venzin
Bocconi University, Milan, Italy

Abstract

We study the *rent-or-buy* variant of the online Steiner forest problem on *node- and edge-weighted* graphs. For n-node graphs with at most $\bar{n}$ nodes of non-zero weight, and at most $\tilde{k}$ different arriving terminal pairs, we obtain the following:

- A *deterministic*, $O(\log n \log \bar{n})$-competitive algorithm against adaptive adversaries. This improves on the previous best, $O(\log^4 n)$-competitive algorithm obtained by the black-box reduction from [5] combined with the previously best deterministic algorithms for the simpler "buy-only" setting.
- A *deterministic*, $O(\bar{n} \log \tilde{k})$-competitive algorithm against adaptive adversaries. This generalizes the $O(\log \tilde{k})$-competitive algorithm for the purely edge-weighted setting from [24].
- A *randomized*, $O(\log \tilde{k} \log \bar{n})$-competitive algorithm against oblivious adversaries. All previous approaches were based on the randomized, black-box reduction from [3] that achieves a $O(\log \tilde{k} \log n)$-competitive ratio when combined with an algorithm for the "buy-only" setting.

Our key technical ingredient is a novel charging scheme to an instance of *online prize-collecting set cover*. This allows us to extend the witness-technique of [24] to the node-weighted setting and obtain refined guarantees with respect to $\bar{n}$, already in the much simpler "buy-only" setting.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online network design, node-weighted Steiner forest, derandomization

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.16

Related Version *Full Version*: https://arxiv.org/abs/2507.08698

1 Introduction

The *Steiner forest* problem is a cornerstone of network design. Given a graph $G = (V, E)$ and pairs of terminals $T \subseteq V \times V$, the goal is to select a minimum weight subgraph that connects each pair. In the *online, node-weighted* setting, there is a cost function $c : V \to \mathbb{R}_{\geq 0}$ on the vertices of the graph, and terminal pairs arrive one-by-one. After every arrival, we need to ensure that the pair is connected. Crucially, decisions cannot be altered in hindsight, i.e. we must *augment* a set of selected vertices so that their induced subgraph connects all arrived pairs so far. The goal is to minimize the total cost of selected vertices. In this paper, we consider the online node-weighted Steiner forest problem in the *rent-or-buy* setting. Given some parameter M, every vertex can either be *rented* at cost c_v, in which case, we may only use it to connect the current pair of terminals, or *bought* at cost $M \cdot c_v$, in which case it may be used indefinitely from then on. The goal is to minimize the total cost of rented vertices and bought vertices. More specifically, the aim is to minimize the *competitive ratio*; the supremum of the ratio between the cost of the online algorithm, and the minimum cost of any feasible solution, for any sequence of terminal pairs.

This problem is centrally situated between several prominent problems in (online) network design. Indeed, the node-weighted setting sits between the *edge-weighted* setting and the *directed* setting[1]. On the other hand, the rent-or-buy setting generalizes over the simpler "buy-only" setting, and serves as a stepping stone for the even more general *buy-at-bulk* setting.[2] The latter is often also referred to as *multicommodity buy-at-bulk*. There have been considerable efforts in devising online algorithms for these problems, both for the Steiner forest and the less general Steiner tree setting. We summarize those algorithmic results in Table 1.

Table 1 State of the art competitive ratios for online network design. The number of vertices in the graph is denoted by n, the number of arriving terminal pairs by k, and the number of *distinct* terminal pairs by $\tilde{k}$. Ratios written with a $\Theta(\cdot)$ are asymptotically tight. Results marked with 🌲 are only valid for the case of single-source (e.g. analog to Steiner tree). Results marked with a dice 🎲 are randomized, the corresponding competitive ratio holds in expectation over the random bits of the algorithm. Results marked with 🙈 only hold against oblivious adversaries, e.g. the adversary cannot observe the algorithms actions and choose the next request adaptively. The algorithm for online directed buy-at-bulk only applies to single-source buy-at-bulk, the version for multicommodity being at least as hard as Label Cover, [13]. It runs in *quasi-polynomial* time 🖩.

	edge-weighted	node-weighted	directed
Buy	$\Theta(\log k)$ [18, 6]	$\Theta(\log k \log n)$ 🎲, 🙈 / $\Theta(\log^2 n)$ [23, 17, 16, 10]	
RoB	$\Theta(\log \tilde{k})$ [3, 24]	$\Theta(\log^2 n)$ 🎲 / $O(\log^4 n)$ [5, 3] + [7, 10]	
BaB	$\Theta(\log \tilde{k})$ 🌲 [15]		$O(\text{poly} \log n)$ 🌲, 🎲, 🙈, 🖩 [14]

The rent-or-buy setting is a common theme in decision making. Problems such as *ski-rental*, *TCP-acknowledgment*, *snoopy caching* or the *parking permit problem*, are prototypical examples, see for instance [19, 20, 22]. More related to the context of network design is the *file replication problem*, which corresponds to a rent-or-buy problem on a tree, see [8, 4]. The first to study the rent-or-buy setting in the context of Steiner problems were Awerbuch, Azar and Bartal, see [3]. In fact, they show that for any online problem that can be cast as a type of *metrical task system*, network design problems being a particular case, there is a simple procedure to adapt an online algorithm to the rent-or-buy setting. More specifically, given any online algorithm that is c-competitive against *adaptive online adversaries* in the buy-only setting, their procedure yields a *randomized* online algorithm for the rent-or-buy setting,

[1] To reduce the edge-weighted to the node-weighted setting, we replace each edge by an unweighted edge subdivided by a node of node-weight equal to the original edge-weight. To reduce the node-weighted to the directed setting, we turn each edge into two (unweighted) directed edges, and split each vertex into v_{in} and v_{out}. For each vertex v, we direct all incoming edges to v_{in}, all outgoing edges from v_{out}, and set the weight of the directed edge v_{in} to v_{out} to be c_v.

[2] The cost of selecting x times an edge/vertex of cost c is $f(x) \cdot c$, for some concave function $f(\cdot)$. The rent-or-buy setting is a special case by setting $f(x) := \min\{x, M\}$. The buy-only setting corresponds to $f(x) := \min\{x, 1\}$.

that is $O(c)$-competitive against adaptive online adversaries.[3] Subsequently, this was further extended to yield a deterministic procedure that yields a $O(c^2)$-competitive deterministic online algorithm, see [5].

Using these two procedures and the corresponding algorithms from the edge-weighted setting, see [18, 6], this directly yields a $\Theta(\log \tilde{k})$-competitive randomized online algorithm for Steiner forest in the rent-or-buy setting, as well as an $O(\log^2 \tilde{k})$-competitive deterministic algorithm. Here, we denote by $\tilde{k}$ the number of *distinct* terminal pairs, as opposed to the buy-only setting, terminal pairs may reappear. Later, a $\Theta(\log \tilde{k})$-competitive deterministic online algorithm was given in [24].

Contrary to the edge-weighted setting, the competitive ratios of all known algorithms for online node-weighted Steiner problems (in the buy-only setting) depend on whether the adversary is adaptive or *oblivious* (non-adaptive). The first approaches were inherently randomized and only hold against oblivious adversaries, [23, 16]. The subsequent algorithm from [16], was then made deterministic at a small loss, by giving a deterministic online algorithm for non-metric facility location, see [7]. Consequently, since any deterministic online algorithm is competitive against adaptive adversaries, combined with the procedure from [3], this yielded the first randomized $O(\log^2 n \log \tilde{k})$-competitive algorithm for node-weighted Steiner forest in the rent-or-buy setting. The most recent online algorithm for node-weighted Steiner forest achieves a competitive ratio $\Theta(\log k \log n)$ against oblivious adversaries. It can be derandomized, resulting in a competitive ratio of $\Theta(\log^2 n)$, see [10]. We note that these two bounds are analogous to (and match) the $\Theta(\log k \log m)$- resp. $\Theta(\log n \log m)$-competitive ratios for online Set Cover in the randomized / deterministic setting[4]. Moreover, these competitive ratios are optimal for polynomial-time algorithms, and they are almost tight even in the absence of computational restrictions [1, 21, 11]. In particular, from the reduction from online Set Cover it follows that it is not possible to obtain a competitive ratio of $O(\log k \log n)$ against adaptive adversaries.

Applying the procedures from [3, 5] to the deterministic algorithm from [10] in a black-box way, we obtain a deterministic $O(\log^4 n)$ online algorithm, as well as a randomized, $O(\log^2 n)$-competitive online algorithm. It is also possible to do the same for the randomized version of this algorithm, yielding an $O(\log \tilde{k} \log n)$-competitive algorithm. However, this needs a more careful argument. We will elaborate on this in Section 1.3.

This leaves quite a gap between the randomized and the deterministic setting. As suggested by the edge-weighted setting, it might indeed be possible to completely match the respective competitive ratios from the buy-only setting. In this paper we confirm this. In fact, we obtain a refined bound which depends on $\bar{n}$, the number of nodes of non-zero weight[5]. Our main result is a deterministic, $O(\log n \log \bar{n})$-competitive online algorithm for Steiner forest in the rent-or-buy setting. This even improves over the respective algorithm in [10] for the buy-only setting. It can be modified to give a deterministic $O(\bar{n} \log \tilde{k})$-competitive algorithm, which generalizes the deterministic algorithm of [24] to the edge- and node-weighted setting. Finally, our last result is a randomized, $O(\log \tilde{k} \log \bar{n})$-competitive algorithm against oblivious adversaries.

[3] In Section 2.3 we give a formal overview of the adversarial settings.

[4] (Online) Set Cover reduces to (online) node-weighted Steiner forest / tree. k corresponds to the number of arriving elements / pairs of arriving terminals, m to the number of vertices in the graph / sets, and n to the number of possible elements / possible pairs of terminals ($n = \binom{m}{2}$). See [10, Section 2.2] for an illustration.

[5] This also applies to nodes of small degree: any node v with constant degree can be replaced by a node with node-weight 0, where we add c_v to its incident edge-weights. This increases the overall cost by at most a constant.

1.1 Our results

We now formally present our results. We consider edge- and node-weighted graphs with n vertices, denote by $\bar{n}$ the number of nodes with non-zero node-weight, and denote by $\tilde{k}$ the number of distinct terminals pairs arriving.

All our results are obtained through the same algorithmic framework, but using three different algorithms for prize-collecting set cover as a subroutine. We present the three resulting results separately.

▶ **Theorem 1.** *There is a deterministic, polynomial-time online algorithm for node-weighted Steiner forest in the rent-or-buy setting which is $O(\log n \log \bar{n})$-competitive against an adaptive adversary. If the terminal pairs are known to come from some subset $T \subseteq V \times V$, the competitive ratio can be improved to $O(\log |T| \log \bar{n})$.*

We consider this to be our main result. All previously known deterministic algorithms use [5], and were a multiplicative factor of $O(\log^2 n)$ worse than the corresponding randomized version. This result closes this gap. We also note that it matches the currently best deterministic algorithm for the buy-only version (e.g. the special case $M = 1$) from [10], and even slightly improves on it, as the dependency is now with respect to $\log \bar{n}$, instead of $\log n$.

The competitive ratio can be slightly improved to $O(\log \tilde{k} \log \bar{n})$, to depend on the number of distinct, *actually* arriving terminals pairs $\tilde{k}$, instead of the number of distinct terminal pairs $|T|$ that *could* arrive. However, to do so, it is necessary to rely on randomization, as already demonstrated in the corresponding lower bounds for online set cover in [2]. This again matches (and slightly improves) the corresponding result from [10] for the buy-only setting.

▶ **Theorem 2.** *There is a randomized, polynomial-time online algorithm for node-weighted Steiner forest in the rent-or-buy setting which is $O(\log \tilde{k} \log \bar{n})$-competitive against an oblivious adversary.*

If one does not care about the logarithmic dependency on $\bar{n}$, it is possible to make this deterministic. The resulting competitive ratio is $O(\bar{n} \cdot \log \tilde{k})$, it depends *linearly* on the number of nodes with non-zero node-weights. This recovers the deterministic, $O(\log \tilde{k})$-competitive algorithm of [24] in the edge-weighted setting.

▶ **Theorem 3.** *There is a deterministic, polynomial-time online algorithm for node-weighted Steiner forest in the rent-or-buy setting which is $O(\bar{n} \cdot \log \tilde{k})$-competitive against an adaptive adversary.*

1.2 Our techniques

On a high-level, we combine the *witness* technique from the edge weighted variant for Rent-or-Buy from [24] with an instance of *prize-collecting set cover*. Before describing our approach, we briefly describe the approach for the edge-weighted setting, as well as the relevant techniques necessary for handling the node-weighted setting.

For the rent-or-buy setting, the difficulty lies in deciding whether it is worthwhile to buy an edge since we can reuse it at least M times in the future, or whether we should rent. In the Ski Rental problem, e.g. Rent-or-Buy on a graph consisting of a single edge, a simple way to decide this and guarantee 2-competitiveness is as follows: we rent M times, then we buy. On a high level, this charging scheme was adapted to the edge-weighted Steiner problems in [24], surprisingly by a simple greedy procedure (though the analysis is quite involved!): if the new request r lies close to M previous requests for which we rented, buy the greedy path

the algorithm selects for r. The cost of the bought path can be charged against these M witnesses and all future requests close to any of them can be routed through the bought path. This ensures there cannot appear any clusters of $\Omega(M)$ rented paths for which it would have been beneficial to just buy once, allowing for a charging scheme against the optimal solution.

Such a greedy approach cannot work in the node-weighted setting, even in the buy-only setting. Because the weights are on the nodes, it is possible for some vertex v to be close to many (eventually) arriving terminals, but, all these terminals are quite far from each other. Indeed, this happens if these terminals have small distance to v, but v has significant weight. Hence, even if we always buy a path for each arriving request, we might still end up with huge clusters of terminals for which an optimal offline solution only buys a single node. Hence, it is necessary to strategically buy certain nodes, which will decrease the distance of future requests. As shown in [23, 17], in the buy-only setting, these nodes can be selected through an instance of online *non-metric facility location*.

There are two main challenges to extend this to the rent-or-buy setting. First, analogous to the charging scheme for Ski-Rental problem, we need to balance the cost of renting and buying for *every* node and edge. Second, the number of requests is no longer bounded by the size of the graph: when we opt to rent, the same request can reappear again and incur an extra cost (unlike in the buy-only setting). Both of these issues cannot be handled by the charging scheme based on standard non-metric facility location. Our main contribution is to introduce a new charging scheme that instead relies on a *prize-collecting* variant of set cover, which itself is a special case of non-metric facility location. This extension to the prize-collecting variant takes care of these issues: It allows us to set up a charging scheme based on [24, 17], that ensures that no more than M *reappearances* of terminals can "cluster", balancing their cost against the cost of the prize-collecting instance of set cover, and naturally handling reappearances of terminals, with the competitive ratio only depending on the number of *distinct* terminals.

Finally, as a by-product of our conceptual simplification of using online (prize-collecting) set cover instead of (prize-collecting) non-metric facility location, our obtained bounds depend on $\bar{n}$, the number of nodes with non-zero node-weights, instead of n, the total number of vertices. This results in a framework that truly generalizes and recovers the best-known results from the edge-weighted setting, without incurring an extra $O(\log n)$ factor.

1.3 Related work

Most relevant to our work is the randomized procedure from [3]. This procedure can convert any algorithm for the buy-only setting into a randomized algorithm for the rent-or-buy setting with only a constant loss in the competitive ratio. It works in a very simple way: each request is passed to the buy-only algorithm with probability $1/M$, and for the remaining requests the algorithm rents. In their original paper, they only claim competitiveness against online adaptive adversaries when the algorithm for the buy-only setting is competitive against online adaptive adversaries. This would rule out using the $O(\log k \log n)$-competitive algorithm from [10], as that algorithm is not competitive against adaptive adversaries. However, a slightly more careful analysis of the result from [3] reveals that the same procedure can be applied, resulting in a randomized, $\Theta(\log \tilde{k} \log n)$-competitive algorithm against oblivious adversaries. This is a weaker adversarial setting than that of an adaptive online adversary. We elaborate on this in the appendix of the full version.

2 Preliminaries

This section is divided in three parts. First, in Section 2.1, we present the necessary background from graph theory. In Section 2.2, we give a formal description of online problems under consideration. Finally, in Section 2.3, we describe the relevant adversarial settings

2.1 Graph Theory

We consider undirected graphs $G = (V, E)$ with a node-weight function $c : V \to \mathbb{R}_{\geq 0}$. This setting also captures edge-weights and is purely for convenience. For any pair of vertices $u, v \in V$, we set $d_G(u, v)$ to be the distance with respect to $c(\cdot)$ between u and v, excluding the weights of u and v. Given some set $A \subset V$, the subgraph induced by A is denoted by $G[A]$, and consists of all vertices in A as well as all edges from E that have both endpoints in A. G/A denotes the graph G where the weight of all nodes in A have been set to 0, and $d_{G/A}(u, v)$ denotes the shortest distance between u and v in G/A, again excluding the weights of u and v. Finally, given some radius $r > 0$ and some vertex u, we define the open ball of radius r around u to be $B(u, r) := \{v \in V \mid d_G(u, v) + c_v \leq r\}$, and its boundary to be $\mathsf{bd}B(u, r) := \{v \in V \mid d_G(u, v) \leq r < d_G(u, v) + c_v\}$. We set $\bar{B}(u, r) := B(u, r) \cup \mathsf{bd}B(u, r)$.

2.2 Online problems

In the online node-weighted Steiner forest problem (NWSF) in the rent-or-buy setting, we are given a node-weighted graph upfront, as well as a parameter M. Then, one-by-one, we receive pairs of terminals $(s_1, t_1), \ldots, (s_k, t_k) \subseteq V \times V$. At each timestep $i \in [k]$, we need to maintain (augment) a set B of bought nodes and select a set of nodes R_i such that (s_i, t_i) is connected in $G[B \cup R_i]$. The total cost is $M \cdot c(B) + \sum_{i \geq 1}^{k} c(R_i)$. The *Steiner tree* problem is defined analogous, except that we need to maintain a *connected* subgraph connecting all the arrived terminals $s_1, \ldots, s_k$; this corresponds to the Steiner forest problem in the setting where the terminal pairs are of the form $(s_1, s_2), (s_1, s_3), \ldots, (s_1, s_k)$.

An instance of the set cover problem (SC) is given by a set system $(X, \mathcal{S}, \mathsf{cost})$, where X is a groundset of elements, $\mathcal{S} \subseteq 2^X$ is a set of subsets of X, and $\mathsf{cost} : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is a cost function. In the online setting, the elements are revealed one-by-one, and we need to ensure it is covered by a set. Sets from the cover cannot be removed, and the goal is to minimize the total cost of the selected sets. This problem has been studied in two settings. In the first, we have *no* knowledge of the set system. Only upon arrival of an element, we learn which sets contain it. In the second setting, we learn the set system $(X, \mathcal{S})$ upfront. The subset of elements in X that needs to be covered is revealed in an online fashion. Deterministic algorithms for online set cover are only meaningful in the second setting. In contrast, randomized algorithms against oblivious adversaries also apply in the first setting.

2.3 Adversarial settings

An online problem is a game between an online algorithm and an adversary. The adversary reveals the input I_{Adv} to some problem piece by piece, after each reveal, the online algorithm must augment their current solution to maintain feasibility. The cost incurred by the online algorithm is compared to that of the adversary, its ratio is referred to as the *competitive ratio*. This notion does depend on the type of adversary, i.e. how the adversary is allowed to generate the sequence I_{Adv} and how it has to service it. We give a brief description of three adversarial settings that we consider in this paper, as well as the corresponding notion of competitiveness. For a more detailed exposition, see [9].

Adaptive adversaries

An adaptive adversary can create the input sequence I_{Adv} adaptively. In each time-step, it selects the next request to be presented to the online algorithm based on previous actions of the online algorithm so far.

Semi-adaptive adversaries

This adversarial setting was introduced recently in [10]. A semi-adaptive adversary commits to a super-instance $\hat{I}_{\mathsf{Adv}}$ of the input sequence. In the online phase, it adaptively selects $I_{\mathsf{Adv}} \subseteq \hat{I}_{\mathsf{Adv}}$ to be presented to the online algorithm. This adversarial setting is relevant in settings where the competitive ratio depends on the size of $\hat{I}_{\mathsf{Adv}}$, such as online set cover against an adaptive adversary.

Oblivious adversaries

An oblivious adversary has to commit to the whole input sequence upfront, and cannot change it based on the actions of the online algorithm.

To define the competitive ratio, we further distinguish between *online* and *offline* adversaries. An online adversary needs to service their own request sequence I_{Adv} in an online fashion, i.e. cannot wait to observe the actions of the online algorithm and service their requests accordingly, while an offline adversary can service their request sequence I_{Adv} after it is finished presenting it to the online algorithm. Denoting by $\mathsf{alg}(I_{\mathsf{Adv}})$ the cost of the online algorithm and by $c(\mathsf{Adv})$ the cost of the (respective) adversary, the online algorithm is said to be γ-competitive if

$$\mathbb{E}_{\mathsf{alg}}[\mathsf{alg}(I_{\mathsf{Adv}})] \leq \gamma \cdot \mathbb{E}_{\mathsf{alg}}[c(\mathsf{Adv})].$$

Note that the expectation of the right-hand-side is also over the randomness of the actions, since, whenever the adversary is adaptive, the actions of the online algorithm may influence future requests. If the adversary is oblivious, the right-hand-side holds deterministically, and if the online algorithm is deterministic, both sides hold deterministically.

3 Online prize-collecting set cover

In this paper, we will make use of the *prize-collecting* variant of the online set cover problem.

This is essentially the online set cover problem, with a small twist: for each arriving element e, there is a penalty p if e is left uncovered. Crucially, elements can reappear (with varying penalties). Formally, such an instance is given by $(\tilde{X}, \mathcal{S}, \mathsf{cost} : \mathcal{S} \to \mathbb{R}_{\geq 0}, p : \tilde{X} \times \mathbb{N} \to \mathbb{R}_{\geq 0})$. Here, we denote the ground set by $\tilde{X}$ to emphasize the difference to the (multi-)set of eventually arriving elements. We will refer to $\tilde{k}$ as the number of distinct arriving elements.

It is important to note that the naive reduction from (online) set cover, i.e. duplicate each element proportionally to its number of reappearances and add singleton sets with corresponding penalty cost, does not yield any meaningful competitive ratio. Indeed, such a reduction creates a number of elements which is proportional to the number of arriving elements, which may be considerably higher than $\tilde{k}$. As an example, for the NWSF problem in the rent-or-buy setting with parameter M, this can be of order $M \cdot \tilde{k}$, which would result in a competitive ratio that depends on M.

3.1 Background on online set cover

All approaches for online set cover are based on the following general framework. Throughout the online phase, a monotonically increasing fractional solution is maintained, that is feasible for all arrived elements. In parallel, a *rounding* scheme is run to convert these fractional values to an actual integral solution, i.e. actually picking sets covering the arrived elements. For the first part, we rely on the following result.

▶ **Theorem 4** (Theorem 3.2, [12]). *There is a deterministic online algorithm that maintains a monotonically increasing fractional solution to the online set cover problem. The algorithm is $O(\log \ell)$-competitive with respect to an optimal fractional solution, where $\ell := \max_{x \in X} |\{S \in \mathcal{S} | x \in S\}|$ is the maximum number of sets that an element is contained in.*

Note that this fractional solution is compared to the best fractional solution in hindsight, and can be maintained deterministically. Hence this holds against an offline adversary in any adversarial model. In contrast, the rounding part does depend on the adversarial setting. Against an oblivious or semi-adaptive adversary, the rounding scheme is randomized and the guarantees hold in expectation. However, in contrast to the deterministic rounding scheme (against adaptive adversaries), the underlying set system does not need to be revealed upfront. We resume these results in the following two theorems.

▶ **Theorem 5** (Theorem 3, [10]). *There is a randomized procedure that rounds a monotonically increasing solution to an instance of online set cover. Against a semi-adaptive adversary that commits to an unrevealed super-instance $\hat{I} = (\hat{X}, \mathcal{S}, \mathsf{cost})$ upfront and adaptively selects $X' \subseteq \hat{X}$ to be presented to the algorithm, we have that:*

$$\mathbb{E}[\mathsf{cost}(\mathsf{alg})] \leq O(\log |\hat{X}|) \cdot \mathbb{E}[\mathsf{val}_{\mathsf{frac}}].$$

Here, $\mathsf{val}_{\mathsf{frac}}$ is the cost of the fractional solution provided to the procedure.

Note that this generalizes the result for online set cover against oblivious adversaries. If the adversary is oblivious to the actions of the rounding procedure, their actions may as well be fixed. Hence, we may assume that $X' = \hat{X}$ and that the right-hand-side holds deterministically. In that case, combined with Theorem 4, this recovers the $O(\log k \log \ell)$-competitive ratio against (offline) oblivious adversaries of [11], where $k = |X'|$ is the number of arriving elements. Against adaptive adversaries, we rely on the following.

▶ **Theorem 6** (Theorem 5.2, [11]). *There is a deterministic online rounding procedure that rounds a monotonically increasing solution to an instance of online set cover, provided the set system $\hat{I} = (\hat{X}, \mathcal{S}, \mathsf{cost})$ is revealed upfront. The adversary adaptively selects a subset of the elements that will arrive. The algorithm satisfies:*

$$\mathbb{E}[\mathsf{cost}(\mathsf{alg})] \leq O(\log |\hat{X}|) \cdot \mathbb{E}[\mathsf{val}_{\mathsf{frac}}].$$

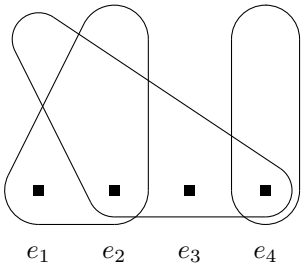
Here, $\mathsf{val}_{\mathsf{frac}}$ is the cost of the fractional solution provided to the procedure.

3.2 Algorithm for prize-collecting online set cover

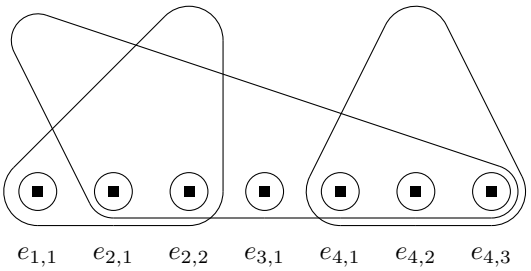
We now give an online algorithm for the prize-collecting variant of online set cover.

We begin by describing an auxiliary instance of set cover $(X, \hat{\mathcal{S}})$. For each reappearance of an element $e \in \tilde{X}$ with penalty p, there is an element $(e, p) \in X$. For each set $S \in \mathcal{S}$, there is a set $\hat{S} \in \hat{\mathcal{S}}$ of same weight, so that $\hat{S}$ contains all copies of the elements contained in S, i.e. $(e, p) \in \hat{S} \implies e \in S$. Finally, for each element corresponding to $(e, p) \in \tilde{X}$, there is a

singleton set $\hat{S}_{(e,p)}$ of weight p in $\hat{\mathcal{S}}$. This construction is illustrated in Figure 1. Clearly, any integral (fractional) solution to this auxiliary instance corresponds to an integral (fractional) solution to the original instance of prize-collecting set cover of same cost and vice-versa.



(a) The prize-collecting set cover instance $(\tilde{X}, \mathcal{S})$.



(b) The auxiliary set cover instance $(X, \hat{\mathcal{S}})$.

Figure 1 In the prize-collecting instance, element e_1, appears once, e_2 twice, e_3 once, and e_4 three times. In the auxiliary instance, each copy is contained in the same sets as the original instance, but is also contained in a singleton set of weight corresponding to its penalty.

We now describe our main algorithm for online prize-collecting set cover on instance $(\tilde{X}, \mathcal{S})$. We start by initializing an (empty) auxiliary instance I_{aux} of online fractional set cover, $(X, \hat{\mathcal{S}})$, for which we will maintain a monotonically increasing fractional solution $\{x_{\hat{S}}\}_{\hat{S} \in \hat{\mathcal{S}}}$ by the algorithm of [12], see Theorem 4. We also initialize an instance $\tilde{I}$ of online set cover on the same set system as $(\tilde{X}, \mathcal{S})$. We will maintain monotonically increasing values $\{x_S\}_{S \in \mathcal{S}}$, for which we will use a rounding scheme for online set cover such as [2, 11, 10] to decide which sets to select for $(\tilde{X}, \mathcal{S})$. Specifically, upon the arrival of an element $(e, p) \in \tilde{X}$, we pass a copy of the element to the auxiliary instance of set cover $(X, \hat{\mathcal{S}})$ as detailed above, and we update our fractional solution. If the fractional value of the singleton set containing (e, p) is at least $1/2$, i.e.

$$x_{\hat{S}_{(e,p)}} > \tfrac{1}{2},$$

we pay the penalty. In parallel, we set $x_S = 2 \cdot x_{\hat{S}}$, i.e. the fractional value of each set $S \in \mathcal{S}$ is twice that of its corresponding set in $\hat{\mathcal{S}}$. We add any sets that the rounding scheme on $(\tilde{X}, \mathcal{S})$ selects based on $\{x_S\}_{S \in \mathcal{S}}$.

▶ **Theorem 7.** *There is a randomized algorithm for online prize-collecting set cover against a semi-adaptive adversary that commits to super-instance $(\tilde{X}, \mathcal{S}, \mathsf{cost})$, but not on the values of the respective penalties. We have that*

$$\mathbb{E}[c(\mathsf{ALG})] \leq O(\log |\tilde{X}| \log |\mathcal{S}|) \cdot \mathbb{E}[c(\mathsf{opt}_{\mathsf{frac}})], \text{ and,}$$

$$c(\mathsf{ALG}_{\mathsf{pen}}) \leq O(\log |\mathcal{S}|) \cdot \mathbb{E}[c(\mathsf{opt}_{\mathsf{frac}})].$$

Here $c(\mathsf{ALG})$ denotes the total cost incurred by the algorithm, $c(\mathsf{ALG}_{\mathsf{pen}})$ the cost incurred on penalties, and $\mathsf{opt}_{\mathsf{frac}}$ an optimum fractional solution to the sub-instance presented to the algorithm. If the set system $(\tilde{X}, \mathcal{S}, \mathsf{cost})$ is revealed upfront, the algorithm can be made deterministic and the above competitive ratio holds deterministically.

Proof. By Theorem 4, the fractional solution $\{x_{\hat{S}}\}_{\hat{S}\in\hat{\mathcal{S}}}$ for the auxiliary instance I_{aux} is $O(\log \ell)$ competitive, where ℓ is the maximum number of sets any element is contained in. Note that this is deterministic and does not require the assumption that the instance be revealed upfront. By construction, ℓ is at most $|\mathcal{S}|+1$. Since we only pay the penalty if the singleton set has weight at least $1/2$, the total cost incurred on penalties is at most

$$O(\log|\mathcal{S}|)\cdot c(\mathsf{opt}_{\mathrm{frac}}).$$

Hence, whenever we do not pay the penalty for element (e,p), the fractional weight of all sets $\hat{S}\ni(e,p)$ except for the singleton set must sum up to at least $1/2$. Since $x_S = 2\cdot x_{\hat{S}}$, the values $\{x_S\}_{S\in\mathcal{S}}$ fractionally cover e. Their fractional cost is $O(\log|\mathcal{S}|)\cdot\mathsf{opt}_{\mathrm{frac}}$. Hence, by Theorem 5 resp. 6, these values are rounded in an online fashion (ensuring e is covered), incurring an $O(\log|\tilde{X}|)$ multiplicative loss. This concludes the proof. ◀

To conclude this section, we describe Theorem 8, a deterministic, $O(|\mathcal{S}|)$-competitive algorithm for online prize-collecting set cover that is 1-competitive with respect to penalties. Specifically, we use a *dual* charging scheme to decide when to pay the penalty and when to buy which sets. The dual is given by (D), the primal, e.g. the fractional relaxation to $(X,\hat{S})$, by (P).

$$\begin{aligned}
&\min \sum_{S\in\hat{\mathcal{S}}} c_{\hat{S}}\cdot x_{\hat{S}} && && \max \sum_{(e,p)\in X} y_{(e,p)}\\
\text{(P)}\quad &\text{s.t.} \sum_{\hat{S}:(e,p)\in\hat{S}} x_{\hat{S}}\geq 1, \quad \forall(e,p)\in X && \text{(D)}\quad &\text{s.t.} \sum_{(e,p)\in\hat{S}} y_{(e,p)}\leq c_{\hat{S}}, \quad \forall\hat{S}\in\hat{\mathcal{S}}\\
&\qquad x_{\hat{S}}\geq 0, \quad \forall\hat{S}\in\hat{\mathcal{S}} && &\qquad y_{(e,p)}\geq 0, \quad \forall(e,p)\in X
\end{aligned}$$

The algorithm is very simple. Whenever an element $e\in\tilde{X}$ with penalty p arrives, we pass (e,p) to the fractional auxiliary instance of set cover described above. We raise the value of the corresponding dual variable $y_{(e,p)}$ until one of the dual constraints (indexed by $\hat{S}\in\hat{\mathcal{S}}$) becomes tight, i.e.

$$\sum_{(e,p)\in\hat{S}} y_{(e,p)} = c_{\hat{S}}.$$

We buy the set in $\mathcal{S}$ corresponding to such a set $\hat{S}$, or pay the penalty if it is the singleton set $\hat{S}_{(e,p)}$.

▶ **Theorem 8.** *If the set system $(\tilde{X},\mathcal{S},\mathsf{cost})$ is not revealed upfront, there is a deterministic, $O(|\mathcal{S}|)$-competitive algorithm for prize collecting set cover. Furthermore, it holds that*

$$c(\mathsf{ALG}_{\mathsf{pen}})\leq c(\mathsf{opt}_{\mathsf{frac}}),$$

where $c(\mathsf{ALG}_{\mathsf{pen}})$ denotes the cost incurred on penalties, and $c(\mathsf{opt}_{\mathsf{frac}})$ denotes the optimal fractional cost to sub-instance presented to the online algorithm.

Proof. Clearly, throughout the algorithm, we maintain a feasible dual solution $y_{(e,p)}$ for (D), and each arriving element is covered. To bound the cost, we observe that by weak duality, (D) $\leq$ (P). Since we only pay the penalty p for element e if $y_{(e,p)}=p$, the total amount spent on penalties (i.e. the singleton sets in $\hat{\mathcal{S}}$) is at most (D). To bound the total cost, we observe that each $y_{(e,p)}$ contributes its value to at most $|\mathcal{S}|+1$ sets, meaning that $c(\mathsf{ALG})\leq(|\mathcal{S}|+1)\cdot(\text{D})\leq O(|S|)\cdot\mathsf{opt}_{\mathsf{frac}}$. ◀

4 Rent-or-Buy for node-weighted Steiner forest

In this part we show Theorems 1, 2 and 3. The algorithm is in Section 4.1, the analysis in Section 4.2.

4.1 The algorithm

Our algorithm relies on an auxiliary instance of online prize-collecting set cover. We describe it in Algorithm 1.

Set-up

We assume that for all arriving terminal pairs, we have that $1 \leq d_{G/A}(s_i, t_i) \leq \tilde{k}^6$, where $\tilde{k}$ is the number of arriving terminal pairs (set $\tilde{k} := |T|$ for Theorem 1), and A is the set of bought nodes. This is without loss of generality, as shown in the full version of this paper. This allows us to group pairs (s_i, t_i) into *layers* $j \in L := \{0, 1, \ldots, 6 \log \tilde{k}\}$ corresponding to $\lfloor \log d_{G/A}(s_i, t_i) + 1 \rfloor$. For ease of presentation we transform the node- and edge-weighted graph into a purely node-weighted graph as follows. Each edge with non-zero edge-weight c_e is subdivided by $\lceil 8 \cdot 1/c_e + 1 \rceil$ nodes, each of node-weight strictly less than $1/8$ and summing exactly to c_e. Note that this transformation is only done for the purpose of the description and analysis of the online algorithm, but is not necessary to construct explicitly.

An instance of online prize-collecting set cover (PCSC)

We describe the set system $(\tilde{X}, \mathcal{S})$. For each node $u \in G$, and each $j \in L := \{0, 1, \ldots, 6 \log \tilde{k}\}$, there is an element $r_{u,j}$. Also, for each node $v \in G$, there is a set S_v with cost $\log(\tilde{k}) \cdot M \cdot c_v$. We define $R_{u,j}$ to be the family of sets S_v with $v \in \mathsf{bd}B(u, 2^{j-5})$ for which $c_v \geq 2^{j-6}$. Element $r_{u,j}$ is contained in all sets in $R_{u,j}$ and its penalty is 2^j. Whenever clear, we drop the prize-collecting and simply refer to this instance as an instance of set cover (SC).

Description of the algorithm

We now give a short overview. We start by initializing an instance of prize-collecting set cover as described above, $|L|$ witness sets $F_0, F_1, \ldots, F_{|L|}$, as well as a counter $y_{w,j} \leftarrow 0$ for each node $w \in G$. Every time a terminal pair (s, t) arrives, we proceed as follows. We compute $d := d_{G/A}(s, t)$, the distance of s to t, and we refer to $j := \lfloor \log d + 1 \rfloor$ as the *layer* the demand pair is on. We always rent the greedy path from s to t (hence satisfying the connectivity requirement), however, we will also buy some nodes. To decide this, we distinguish between two cases, depending on whether s or t is *uncovered* on layer j: s is said to be uncovered on layer j, if no witness set on layer j containing s has been added to the set cover, and, for all vertices v in the open ball of radius 2^{j-5} centered at s, the counter $y_{v,j}$ is strictly less than M; e.g.

$$R_{s,j} \cap F_j = \emptyset, \text{ and } , \forall w \in B(s, 2^{j-5}) : y_{w,j} < M.$$

If this holds, we pass s (or t, whatever is uncovered) to the auxiliary instance of set cover.

If the instance of set cover adds a new set to the cover, we buy the corresponding node and add it to F_j. For all nodes $w \in B(s, 2^{j-5})$ we increase the counter $y_{w,j}$ by one. If this does not hold, i.e. both s and t are covered, we proceed as follows. We pick two nodes $w_s, w_t \in F_j$ on layer j, where w_s (resp. w_t) lies in the closed ball of radius 2^{j-3} around s (resp. t). If this set is empty, we add s (resp. t) to the witness set F_j and set $w_s = s$ (resp. $w_t = t$). We then buy the shortest path between w_s and w_t in G/A.

Size of set system

From the description of the algorithm, it is clear that the only elements $r_{u,j}$ that are released to the instance of PCSC, correspond to some terminal u in the original graph. Hence, the number of arriving elements of $\tilde{X}$ is at most $O(\tilde{k} \cdot \log \tilde{k})$, and the number of elements that could potentially arrive is at most $O(|T| \cdot \log |T|)$. To see that we can bound the number of sets by $\bar{n}$, note that the nodes corresponding to non-zero edge-weights in the original graph have node-weights strictly less than $1/8$ and do not contain any elements. As such, they are irrelevant and can be disregarded. All remaining sets correspond to one of the $\bar{n}$ original nodes with non-zero node-weight.

Algorithm 1 Node-weighted Rent-Or-Buy Steiner Forest.

1: $F_j \leftarrow \emptyset$ for all $j \in L$
2: Initialize PCSC, an algorithm for online prize-collecting set cover.
3: **when** (s_i, t_i) arrives **do**
4: $\quad d \leftarrow$ the distance from s_i to t_i in G/A
5: $\quad j \leftarrow \lfloor \log(d) + 1 \rfloor$
6: $\quad$ **if** $\exists\ v \in \{s_i, t_i\}$ that is uncovered on layer j **then**
7: $\quad\quad$ Pass $r_{v,j}$ to PCSC.
8: $\quad\quad$ If $r_{v,j}$ is covered by a newly added set S_u, add u to A and all F_j with $c_u \geq 2^{j-6}$
9: $\quad\quad$ For all $w \in B(v, 2^{j-5})$, set $y_{v,j} \leftarrow y_{v,j} + 1$.
10: $\quad$ **else**
11: $\quad\quad$ If $F_j \cap \bar{B}(s_i, 2^{j-3})$ is empty add s_i to F_j.
12: $\quad\quad$ If $F_j \cap \bar{B}(t_i, 2^{j-3})$ is empty add t_i to F_j.
13: $\quad\quad$ Let $v \in F_j \cap \bar{B}(s_i, 2^{j-3})$ and $w \in F_j \cap \bar{B}(t_i, 2^{j-3})$.
14: $\quad\quad$ Buy v and w and the shortest path between them in G/A.
15: Rent the shortest path between s_i and t_i in G/A.

4.2 The analysis

In this section we prove Theorems 1, 2 and 3. To do so, we are going to show that the cost of Algorithm 1 can be charged against the optimum cost of the instance of prize-collecting set cover.

We will consider the if-case and the else-case separately. Let C_{if} be the cost incurred in iterations that execute the if-case and C_{else} be the cost incurred in the else-case. Let C_{SC} be the cost of the constructed solution to the prize-collecting set cover instance and let $C_{\mathsf{SC,pen}}$ and $C_{\mathsf{SC,sets}}$ be the penalty costs and the costs for the added sets in the solution, respectively. Note that $C_{\mathsf{SC}} = C_{\mathsf{SC,pen}} + C_{\mathsf{SC,sets}}$. We rely on the following four claims.

▷ Claim 9. $C_{\mathsf{else}} \leq 6M \sum_j 2^j |F_j|$.

Proof. The total cost incurred in an iteration in which the else-case is executed can be upper bounded by $3M \cdot 2^j$: at most $2M \cdot 2^j$ is incurred on Line 14, and at most 2^j is incurred on Line 15. So we can prove the above claim by simply bounding the number of iterations in which the else-case is executed for each j by $2|F_j|$. Each such iterations falls into one of the following categories:

- If s_i or t_i are added to F_j in Lines 11 and 12, then $|F_j|$ increases by at least 1 in this iteration. Clearly, this can happen at most $|F_j|$ times.

- Otherwise, some $v \in F_j \cap \bar{B}(s_i, 2^{j-3})$ and $w \in F_j \cap \bar{B}(t_i, 2^{j-3})$ are chosen. Now we claim that no path connecting v and w has been bought so far, i.e. v and w are not connected in G/A. If this was the case, then the distance d between s_i and t_i in G/A would be at most $2^{j-3} + 2^{j-3} = 2^{j-2}$, contradicting the fact that $j = \lfloor \log(d) + 1 \rfloor$. Since v and w become connected in this iteration, the number of components in A that contain a node in F_j decreases by at least 1. Since the number of such components is at most $|F_j|$, this can happen at most $|F_j|$ times.

Since the number of iterations for each of the categories is at most $|F_j|$, the total number of iterations in which the else-case is executed for each j is at most $2|F_j|$. This implies the claim. ◁

▷ Claim 10. $M \sum_j 2^j |F_j| \leq 128(C_{\mathsf{SC,pen}} + \frac{C_{\mathsf{SC,sets}}}{\log \tilde{k}})$.

Proof. Let $F_{j,1}$ be the set of nodes in F_j that were added on Line 8 and let $F_{j,2}$ be the set of nodes in F_j that were added on Lines 11 and 12. Note that $F_j = F_{j,1} \cup F_{j,2}$. We will show that for each time that $|F_{j,1}|$ or $|F_{j,2}|$ increases by 1, the right-hand side of the inequality increases by at least $M2^j$.

- When some new set S_u for $u \in R_{v,j}$ is added to the set cover solution, $|F_{j,1}|$ increases by one for all j with $c_v \geq 2^{j-6}$. Hence, the left-hand side of our claimed inequality grows by at most $M \sum_{i=0}^{\infty} 2^{6-i} \cdot c_v \leq 128M \cdot c_v$, while the right-hand side increases by $128M \cdot c_v$.
- When a node v is added to $F_{j,2}$ (on Lines 11 or 12), node v can not have been uncovered on layer j. This means that either there exists a $z \in R_{v,j} \cap F_j \neq \emptyset$ or there is a $w \in B(v, 2^{j-5}) : \; y_{w,j} \geq M$. However, in the former case, $z \in F_j \cap \bar{B}(v, 2^{j-5})$, in which case v would not be added to F_j. So, there must be some $w \in B(v, 2^{j-5})$ such that $y_{w,j} \geq M$. Call $y_{w,j}$ a *witness* for v.

  We will now charge the increase in $|F_{j,2}|$ to all iterations in which $y_{w,j}$ was increased. Note that in each such iteration, either a set of cost at least $\log(k)M \cdot 2^{j-6}$ is added to the set cover solution or a penalty of 2^j is paid. So in M such iterations, the right-hand side of our claimed inequality increases by at least $M \cdot 2^j$.

  However, we need to be careful that an iteration in which $y_{w,j}$ is increased is not charged by multiple nodes v. We will show that this is not the case. Consider an iteration in which $y_{w,j}$ is increased for all $w \in B(v, 2^{j-5})$. Suppose that $y_{w_1,j}$ and $y_{w_2,j}$ for $w_1, w_2 \in B(v, 2^{j-5})$ are witnesses for v and v' respectively with $v \neq v'$. Assume that v gets added to F_j before v'. By the definition of the witnesses, we have $d(w_1, v) \leq 2^{j-5}$ and $d(w_2, v') \leq 2^{j-5}$. However, since $y_{w_1,j}$ and $y_{w_2,j}$ are both increased in the same iteration, we must have $d(w_1, w_2) \leq 2 \cdot 2^{j-5}$. This implies that $d(v, v') \leq d(v, w_1) + d(w_1, w_2) + d(w_2, v') \leq 4 \cdot 2^{j-5} = 2^{j-3}$, which contradicts the fact that v' is added to F_j even though v is already in F_j. Hence, each iteration in which $y_{w,j}$ is increased is charged by at most one node v. ◁

▷ Claim 11. $C_{\mathsf{if}} \leq O(C_{\mathsf{SC,pen}} + \frac{C_{\mathsf{SC,sets}}}{\log \tilde{k}})$.

Proof. We will show that each iteration the right-hand side increases faster than the left-hand side. Whenever the if-case is executed there are two cases:

- For iterations in which new set S_v is added to the set cover solution, $C_{\mathsf{SC,sets}}$ increases by at least $\log \tilde{k} \cdot M \cdot c_v \geq \log \tilde{k} \cdot M \cdot 2^{j-6}$. To buy v, a cost of $M \cdot c_v$ is incurred. A cost of at most $d \leq 2^j$ is incurred on Line 15 for renting the path between s_i and t_i. So, C_{if} increases by at most $M \cdot c_v + 2^j$, which matches the increase in $O(\frac{C_{\mathsf{SC,sets}}}{\log \tilde{k}})$ on the right-hand-side.

- In all other iterations in which the if-case is executed, $r_{v,j}$ is not covered and a penalty of 2^j is incurred in $C_{\mathsf{SC,pen}}$. The only cost incurred in C_{if} is for renting a path between s_i and t_i on Line 15 of cost at most 2^j. ◁

▷ Claim 12. $\mathsf{opt}_{\mathsf{SC}} \leq O(\log \tilde{k}) \cdot \mathsf{opt}_{\mathsf{RoB}}$.

Proof. Consider opt, an optimal solution to the given Rent-Or-Buy problem, and denote A to be the set of nodes that are bought in opt. We will construct a solution to prize-collecting set cover of cost at most $O(\log \tilde{k})$ times the cost of this optimal solution. To do so, we let the set cover consist of all sets s_v corresponding to nodes v that are bought in opt. This incurs a set cost of $\sum_{v \in A} M \log(\tilde{k}) c_v \leq O(\log \tilde{k}) \cdot \mathsf{opt}$. It remains to show the same bound on the penalty costs.

Consider such an arriving element $r_{v,j}$ that is not covered by the set cover. Let $v_1, v_2, \ldots, v_r$ be a sequence of nodes that connect $v = v_1$ to a node in $v_r \in \mathsf{bd}B(v, 2^{j-5})$ in the optimal Rent-Or-Buy solution. Suppose that $c_{v_r} \geq 2^{j-6}$. Then we must have $v_r \notin A$, since otherwise, we would have added s_{v_r} in our set cover solution and $r_{v,j}$ would have been covered. So, renting v_r incurred a cost of at least 2^{j-6} in $\mathsf{opt}_{\mathsf{RoB}}$. So we can charge the penalty of 2^j incurred in $\mathsf{opt}_{\mathsf{SC}}$ to this cost.

On the other hand, if $c_{v_r} < 2^{j-6}$, then we will charge $\frac{c_{v_i}}{\log \tilde{k}}$ of the penalty incurred by the arrival of $r_{v,j}$ to the cost paid for buying or renting v_i in the optimal Rent-Or-Buy solution for $i \in \{2, r-1\}$. So, in this case we are charging $\sum_{i=2}^{r-1} \frac{c_{v_i}}{\log \tilde{k}} \geq \frac{2^{j-5} - 2^{j-6}}{\log \tilde{k}} = \frac{2^{j-6}}{\log \tilde{k}}$. This is indeed only a factor of $2^6 \log \tilde{k}$ less than the connection cost of $r_{v,j}$.

The only thing to check is that the cost for buying or renting a node in the optimal Rent-Or-Buy solution is larger than the charged cost. Nodes that are not bought in the optimal solution have to be rented at cost c_v each time they are used. So this rental cost is clearly at least as large as the charged cost.

For a node v that is bought in the optimal rent-or-buy solution, observe that $y_{v,j}$ is increased for some j each time that $\frac{c_{v_i}}{\log \tilde{k}}$ is charged to it. Note that $y_{v,j}$ never becomes larger than M. So, the total charged cost for v is at most $M \frac{c_v}{\log \tilde{k}}$ for each j. Hence, the total charged cost for v is at most $M \cdot c_v$, which is exactly the cost of buying v in the optimal solution. ◁

Proof of Theorems 1, 2 and 3. By Claims 9, 10 and 11, the cost of incurred by online algorithm is at most $O(C_{\mathsf{SC,pen}} + \frac{C_{\mathsf{SC,sets}}}{\log \tilde{k}})$. By Claim 12 we know that the optimal solution to the prize-collecting set cover instance is at most $O(\log \tilde{k})$ times the optimal solution to the Rent-Or-Buy Steiner Forest problem.

Plugging in the algorithms for prize-collecting set cover from Theorem 7 in our algorithm, gives us $C_{\mathsf{SC,pen}} \leq O(\log |\mathcal{S}|)\mathsf{opt}_{\mathsf{SC}}$ and $C_{\mathsf{SC,sets}} \leq O(\log |\hat{X}| \log |\mathcal{S}|)\mathsf{opt}_{\mathsf{SC}}$. Hence, the cost of the online algorithm is at most:

$$\begin{aligned} C_{\mathsf{SC,pen}} + \frac{C_{\mathsf{SC,sets}}}{\log \tilde{k}} &\leq O\left(\log |\mathcal{S}| + \frac{\log |\hat{X}| \log |\mathcal{S}|}{\log \tilde{k}}\right) \cdot \mathsf{opt}_{\mathsf{SC}} \\ &\leq O(\log |\mathcal{S}| (\log \tilde{k} + \log |\hat{X}|)) \cdot \mathsf{opt}_{\mathsf{RoB}}. \end{aligned}$$

Since $|\tilde{X}| = O(\tilde{k} \log \tilde{k})$ (recall that we set $\tilde{k} := |T|$ when we instantiate Algorithm 1 with the deterministic algorithm from Theorem 7) and $|\mathcal{S}| = \bar{n}$, Theorems 1 and 2 follow. Similarly, plugging in the deterministic algorithm from Theorem 8 gives us a competitive ratio of $O(\bar{n} \log \tilde{k})$. This proves Theorem 3. ◀

References

1 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. A general approach to online network optimization problems. *ACM Trans. Algorithms*, 2:640–660, 2006. doi:10.1145/1198513.1198522.

2 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. The online set cover problem. *SIAM J. Comput.*, 39:361–370, 2009. Preliminary version in STOC'03. doi:10.1137/060661946.

3 Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. *Theoretical Computer Science*, 324:313–324, 2004. Preliminary version in SODA'96. doi:10.1016/j.tcs.2004.05.021.

4 Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358, 1995. doi:10.1006/jcss.1995.1073.

5 Yair Bartal, Moses Charikar, and Piotr Indyk. On page migration and other relaxed task systems. *Theoretical Computer Science*, 268(1):43–66, 2001. On-line Algorithms '98. doi:10.1016/S0304-3975(00)00259-0.

6 Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In *STOC*, 1997. doi:10.1145/258533.258618.

7 Marcin Bienkowski, Björn Feldkord, and Paweł Schmidt. A Nearly Optimal Deterministic Online Algorithm for Non-Metric Facility Location. In *STACS*, 2021. doi:10.4230/LIPIcs.STACS.2021.14.

8 D. Black and D. Sleator. Competitive algorithms for replication and migration problems. Technical report, Carnegie-Mellon, 1989.

9 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

10 Sander Borst, Marek Eliáš, and Moritz Venzin. Stronger adversaries grow cheaper forests: online node-weighted steiner problems. In *SODA*, 2025. doi:10.1137/1.9781611978322.129.

11 Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal–dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3:93–263, 2009. doi:10.1561/0400000024.

12 Niv Buchbinder and Joseph (Seffi) Naor. Online Primal-Dual Algorithms for Covering and Packing. *Mathematics of Operations Research*, 34(2):270–286, May 2009. doi:10.1287/moor.1080.0363.

13 Yevgeniy Dodis and Sanjeev Khanna. Design networks with bounded pairwise distance. In *STOC*, 1999. doi:10.1145/301250.301447.

14 Alina Ene, Deeparnab Chakrabarty, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online buy-at-bulk network design. In *FOCS*, 2015. doi:10.1109/FOCS.2015.40.

15 Anupam Gupta, R. Ravi, Kunal Talwar, and Seeun William Umboh. Last but not least: Online spanners for buy-at-bulk. In *SODA*, 2017. doi:10.1137/1.9781611974782.38.

16 MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Near-optimal online algorithms for prize-collecting steiner problems. In *ICALP*, 2014. doi:10.1007/978-3-662-43948-7_48.

17 MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online Node-weighted Steiner Forest and Extensions via Disk Paintings. *SIAM J. Comput.*, 46:911–935, 2017. doi:10.1137/14098692X.

18 Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4:369–384, 1991. doi:10.1137/0404033.

19 Anna R. Karlin, Claire Kenyon, and Dana Randall. Dynamic tcp acknowledgement and other stories about $e/(e-1)$. In *STOC*, 2001. doi:10.1145/380752.380845.

20 Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan Owicki. Competitive randomized algorithms for non-uniform problems. In *SODA*, pages 301–309, 1990. URL: http://dl.acm.org/citation.cfm?id=320176.320216.

21 Simon Korman. On the use of randomization in the online set cover problem. Master's thesis, The Weizmann Institute of Science, 2004.

22 A. Meyerson. The parking permit problem. In *FOCS*, 2005. doi:10.1109/SFCS.2005.72.

23 Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted steiner tree and related problems. In *FOCS*, 2011. doi:10.1109/FOCS.2011.65.

24 Seeun Umboh. Online network design algorithms via hierarchical decompositions. In *SODA*, 2015. doi:10.1137/1.9781611973730.91.