# The $(h, k)$-Server Problem on Bounded Depth Trees

NIKHIL BANSAL, CWI and TU Eindhoven, The Netherlands
MAREK ELIÁŠ, EPFL, Switzerland
ŁUKASZ JEŻ, University of Wrocław, Poland
GRIGORIOS KOUMOUTSOS, Université Libre de Bruxelles, Belgium

We study the $k$-server problem in the resource augmentation setting, i.e., when the performance of the online algorithm with $k$ servers is compared to the offline optimal solution with $h \leq k$ servers. The problem is very poorly understood beyond uniform metrics. For this special case, the classic $k$-server algorithms are roughly $(1 + 1/\epsilon)$-competitive when $k = (1 + \epsilon)h$, for any $\epsilon > 0$. Surprisingly, however, no $o(h)$-competitive algorithm is known even for HSTs of depth 2 and even when $k/h$ is arbitrarily large.

We obtain several new results for the problem. First, we show that the known $k$-server algorithms do not work even on very simple metrics. In particular, the Double Coverage algorithm has competitive ratio $\Omega(h)$ irrespective of the value of $k$, even for depth-2 HSTs. Similarly, the Work Function Algorithm, which is believed to be optimal for all metric spaces when $k = h$, has competitive ratio $\Omega(h)$ on depth-3 HSTs even if $k = 2h$. Our main result is a new algorithm that is $O(1)$-competitive for constant depth trees, whenever $k = (1 + \epsilon)h$ for any $\epsilon > 0$. Finally, we give a general lower bound that any deterministic online algorithm has competitive ratio at least 2.4 even for depth-2 HSTs and when $k/h$ is arbitrarily large. This gives a surprising qualitative separation between uniform metrics and depth-2 HSTs for the $(h, k)$-server problem.

CCS Concepts: • **Theory of computation → K-server algorithms**;

Additional Key Words and Phrases: $k$-server problem, online algorithms, resource augmentation, competitive analysis

**ACM Reference format:**
Nikhil Bansal, Marek Eliáš, Łukasz Jeż, and Grigorios Koumoutsos. 2019. The $(h, k)$-Server Problem on Bounded Depth Trees. *ACM Trans. Algorithms* 15, 2, Article 28 (February 2019), 26 pages.
https://doi.org/10.1145/3301314



A preliminary version of this article appeared in the Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) 2017. This work was carried out while Marek Eliáš and Grigorios Koumoutsos were PhD students at TU Eindhoven and Łukasz Jeż was postdoctoral fellow at TU Eindhoven.
The first author was supported by NWO Vidi Grant No. 639.022.211 and ERC consolidator Grant No. 617951.
The second author was supported by NWO Vidi Grant No. 639.022.211.
The third author was supported by NWO Vidi Grant No. 639.022.211 and Polish National Science Centre Grant No. 2016/22/E/ST6/00499, 2017-2022.
The last author was supported by ERC consolidator Grant No. 617951.
Authors' addresses: N. Bansal, CWI and TU Eindhoven, P.O. Box 94079, The Netherlands; email: bansal@gmail.com; M. Eliáš, EPFL, School of Computer and Communication Sciences, Building INJ (INJ132), Station 14, CH-1015, Lausanne, Switzerland; email: elias@ba30.eu; Ł. Jeż, University of Wrocław, Institute of Computer Science, ul. Joliot-Curie 15, 50-383, Wrocław, Poland; email: lje@cs.uni.wroc.pl; G. Koumoutsos, Université Libre de Bruxelles, Département d'Informatique, ULB CP 212, Bvd. du Triomphe, Brussels, 1050, Belgium; email: gregkoumoutsos@gmail.com.

1549-6325/2019/02-ART28 $15.00
https://doi.org/10.1145/3301314

## 1 INTRODUCTION

The classic $k$-server problem, introduced by Manasse et al. (1990), is a broad generalization of various online problems and is defined as follows. There are $k$ servers that reside on some points of a given metric space. At each step, a request arrives at some point of the metric space and must be served by moving some server to that point. The goal is to minimize the total distance traveled by the servers.

In this article, we study the resource augmentation setting of the problem, also known as the "weak adversary" model (Koutsoupias 1999), where the online algorithm has $k$ servers, but its performance is compared to a "weak" offline optimum with $h \leq k$ servers. We will refer to this as the $(h, k)$-server problem. Our motivation is twofold. Typically, the resource augmentation setting gives a much more refined view of the problem and allows one to bypass overly pessimistic worst case bounds; see, e.g., Kalyanasundaram and Pruhs (2000). Second, as we discuss below, the $(h, k)$-server problem is much less understood than the $k$-server problem and seems much more intriguing.

### 1.1 Previous Work

The $k$-server problem has been extensively studied; here, we will focus only on deterministic algorithms. It is well-known that no algorithm can be better than $k$-competitive for any metric space on more than $k$ points (Manasse et al. 1990). In their breakthrough result, Koutsoupias and Papadimitriou (1995) showed that the Work Function Algorithm (WFA) is $(2k - 1)$-competitive in any metric space. For special metrics such as uniform metrics,[1] the line, and trees, tight $k$-competitive algorithms are known (cf. Borodin and El-Yaniv (1998)). It is widely believed that a $k$-competitive algorithm exists for every metric space (the celebrated $k$-server conjecture), and it is also plausible that the WFA achieves this guarantee. Qualitatively, this means that general metrics are believed to be no harder than the simplest possible case of uniform metrics.

**The $(h, k)$-server problem.** Much less is known for the $(h, k)$-server problem. In their seminal paper, Sleator and Tarjan (1985) gave several $\frac{k}{k-h+1}$-competitive algorithms for uniform metrics and also showed that this is the best possible ratio. This bound was later extended to the weighted star metric (weighted paging) (Young 1994). Note that this guarantee equals $k$ for $k = h$ (the usual $k$-server setting), and tends to 1 as $k/h$ approaches infinity. In particular, for $k = 2h$, this is smaller than 2.

It might seem natural to conjecture that, analogously to the $k$-server case, general metrics are no harder than the uniform metrics, and hence that $k/(k - h + 1)$ is the right bound for the $(h, k)$-server problem in all metrics. However, surprisingly, Bar-Noy and Schieber (cf. Borodin and El-Yaniv (1998), p. 175) showed this to be false: In the line metric, for $h = 2$, no deterministic algorithm can be better than 2-competitive, regardless of the value of $k$. This is the best known lower bound for the general $(h, k)$-server problem.

However, the best known upper bound is $2h$, even when $k/h \to \infty$. In particular, Koutsoupias (1999) showed that the WFA with $k$ servers is (about) $2h$-competitive against an offline optimum with $h$ servers. Note that one way to achieve a guarantee of $2h - 1$ is simply to disable the $k - h$ extra online servers and use WFA with $h$ servers only. The interesting thing about the result of Koutsoupias (1999) is that the online algorithm does not know $h$ and is $2h$-competitive simultaneously for every $h \leq k$. But, even if we ignore this issue of whether the online algorithm knows $h$ or not, no guarantee better than $h$ is known, even for very special metrics such as depth-2 HSTs or the line, and even when $k/h \to \infty$.

[1] The $k$-server problem in uniform metrics is equivalent to the paging problem.

### 1.2 Our Results

Motivated by the huge gap between the known lower and upper bounds even for very simple metrics, we consider bounded-depth trees and HSTs (defined formally in Section 1.3).

We first show very strong lower bounds on all the previously known algorithms (beyond uniform metrics), specifically the Double Coverage (DC) algorithm of Chrobak et al. (1991) and Chrobak and Larmore (1991) and the WFA. This is perhaps surprising, because, for the $k$-server problem, DC attains the optimal competitive ratio in trees and WFA is believed to attain the optimal competitive ratio in all metrics.

Theorem 1.1. *The competitive ratio of DC in depth-2 HSTs is $\Omega(h)$, even when $k/h \to \infty$.*

For the WFA, we present the following lower bound.

Theorem 1.2. *The competitive ratio of the WFA is at least $h + 1/3$ in a depth-3 HST for $k = 2h$.*

This lower bound can be extended to the line metric. Note that for the line it is known that the WFA is $h$-competitive for $k = h$ (Bartal and Koutsoupias 2004), while our lower bound for $k = 2h$ is strictly larger than $h$. In other words, our result shows that there exist instances for which the WFA performs strictly worse with $2h$ servers than using $h$ servers! A similar fact was recently observed for the Double Coverage (DC) algorithm in Bansal et al. (2018), where it was shown that its competitive ratio is $\frac{k(h+1)}{k+1}$, which equals $h$ for $k = h$ and tends to $h + 1$ as $k$ grows. Interestingly, our lower bound matches the upper bound $(h + 1)\mathrm{OPT}_h - \mathrm{OPT}_k$ implied by results of Bartal and Koutsoupias (2004) and Koutsoupias (1999) for the WFA in the line. We describe the details in Appendix B.

Our main result is the first $o(h)$-competitive algorithm for depth-$d$ trees.

Theorem 1.3. *There is an algorithm that is $O_d(1)$-competitive on any depth-$d$ tree, whenever $k = \delta h$ for $\delta > 1$. More precisely, its competitive ratio is $O(d \cdot 2^d)$ for $\delta \in [2^d, +\infty)$, and $O(d \cdot (\frac{\delta^{1/d}}{\delta^{1/d}-1})^d)$ for $\delta \in (1, 2^d)$. If $\delta$ is very small, i.e., $\delta = 1 + \epsilon$ for $0 < \epsilon \leq 1$, then the latter bound becomes $O(d \cdot (2d/\epsilon)^d)$.*

Note that the competitive ratio of our algorithm when $k \gg h$ is $O(1)$ for $d = O(1)$, and $o(h)$ for any $d = o(\log h)$. The algorithm is designed to overcome the drawbacks of DC and WFA, and can be viewed as a more aggressive and cost-sensitive version of DC. It moves the servers more aggressively at non-uniform speeds towards the region of the current request, giving a higher speed to a server located in a region containing many servers. It does not require the knowledge of $h$, and is simultaneously competitive against all $h$ strictly smaller than $k$.

Finally, we give an improved lower bound. Bar-Noy and Schieber (cf. Borodin and El-Yaniv (1998), p. 175) showed that there is no better than 2-competitive algorithm for the $(h, k)$-server problem in the line. Our next result shows that the competitive ratio of the $(h, k)$-server problem is least 2.4.

Theorem 1.4. *There is no 2.4-competitive deterministic algorithm for trees of depth 2, even when $k/h \to \infty$, provided that $h$ is larger than some constant independent of $k$.*

This shows that depth-2 trees are qualitatively quite different from depth-1 trees (same as weighted star graphs), which allow a ratio $k/(k - h + 1)$. We have not tried to optimize the constant 2.4 above, but computer experiments suggest that the bound can be improved to about 2.88. Table 1 summarizes the best known competitive ratios for the $(h, k)$-server problem for all metrics considered in the literature, including our contribution.

*Recent Developments.* Since the initial announcement of this work (Bansal et al. 2017), Coester et al. (2017) have studied another variant of the $k$-server problem, called the *infinite server problem*.

Table 1. Overview of the Best Known Competitive Ratios for the $(h, k)$ Server Problem on Certain Metric Spaces, for Various Values of $h$ and $k$

| $k$ | Depth-1 trees | Depth-d trees | Line | General |
|---|---|---|---|---|
| $h$ | $h$ (DC, WFA) | $h$ (DC) | $h$ (DC, WFA) | $2h-1$ (WFA) |
| $(1+\epsilon)\cdot h$ | $\frac{1+\epsilon}{\epsilon+\frac{1}{h}}$ (DC) | $O(d\cdot(2d/\epsilon)^d)$ | $(h+1)\cdot\frac{1}{1+\frac{1}{(1+\epsilon)h}}$ (DC) | $2h$ (WFA) |
| $2h$ | $2\cdot\frac{h}{h+1}\leq 2$ (DC) | $O(d\cdot(2d)^d)$ | $(h+1)\cdot\frac{1}{1+\frac{1}{2h}}$ (DC) | $2h$ (WFA) |
| $\to\infty$ | $\to 1$ (DC) | $O(d\cdot 2^d)$ | $h+1$ (DC) | $2h$ (WFA) |

The algorithms achieving the best known competitive ratio are indicated in the parentheses. The competitive ratios obtained in this work are shown in red.

They show a surprising tight connection between this problem and the $(h, k)$-server problem. Using this connection, they show a lower bound of 3.146 for the competitive ratio of any deterministic algorithm for the $(h, k)$-server problem in the line (for sufficiently large $h$), even if $k/h \to \infty$.

### 1.3 Notation and Preliminaries

In the $(h, k)$-setting, we define the competitive ratio as follows. An online algorithm $ALG$ is $R$-competitive in metric $M$, if there exists a constant $\alpha$ such that $ALG_k(I) \leq R \cdot OPT_h(I) + \alpha$ holds for any finite request sequence $I$ of points in $M$. Here, $ALG_k(I)$ denotes the cost of $ALG$ serving $I$ with $k$ servers and $OPT_h(I)$ denotes the optimal cost to serve $I$ with $h$ servers. An excellent reference on competitive analysis is Borodin and El-Yaniv (1998).

A *depth-d tree* is an edge-weighted rooted tree with each leaf at depth exactly $d$. In the $(h, k)$-server problem in a depth-$d$ tree, weights of edges are interpreted as distances and the requests arrive only at leaves. The distance between two leaves is defined as the distance in the underlying tree. A depth-$d$ HST is a depth-$d$ tree with the additional property that the distances decrease geometrically away from the root (see, e.g., Bartal (1996)). We will first present our algorithm for general depth-$d$ trees (without the HST requirement), and later show how to (easily) extend it to arbitrary trees with bounded diameter, where requests are also allowed in the internal nodes.

**DC Algorithm for Trees.** Our algorithm can be viewed as a non-uniform version of the DC algorithm for trees (Chrobak et al. 1991; Chrobak and Larmore 1991), which works as follows. When a request arrives at a vertex $v$ of the tree, all the servers adjacent to $v$ move towards it along the edges at the same speed until one eventually arrives at $v$. Here, a server $s$ is *adjacent* to a point $x$ if there is no other server on the (unique) path from $s$ to $x$. If multiple servers are located at a single point, then only one of them is chosen arbitrarily.

**Work Function Algorithm.** Consider a request sequence $\sigma = r_1, r_2, \ldots, r_m$. For each $i = 1, \ldots, m$, let $w_i(A)$ denote the optimal cost to serve requests $r_1, r_2, \ldots, r_i$ and end up in the configuration $A$, which is specified by the set of locations of the servers. The function $w_i$ is called *work function*. The Work Function Algorithm (WFA) decides its moves depending on the values of the work function. Specifically, if the algorithm is in a configuration $A$ and a request $r_i \notin A$ arrives, it moves to a configuration $X$ such that $r_i \in X$ and $w_i(X) + d(A, X)$ is minimized. For more background on the WFA, see Borodin and El-Yaniv (1998).

### 1.4 Organization

In Section 2, we describe the lower bound for the DC in depth-2 HSTs. The shortcomings of the DC might help the reader to understand the motivation behind the design of our algorithm for depth-$d$ trees, which we describe in Section 3. Its extension to the bounded-diameter trees can be found in Appendix A.2. In Section 4, we describe the general lower bound (Theorem 1.4) and the

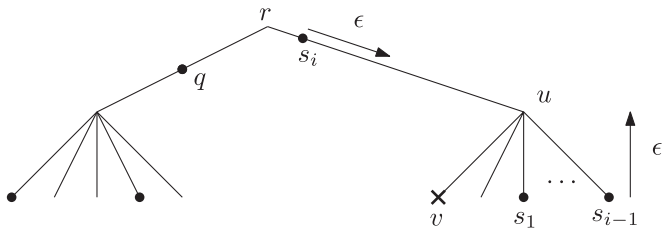


Fig. 1. Move of DC during Step $i$. Servers $s_1, \ldots, s_{i-1}$ are moving towards $u$ by distance $\epsilon$ and $s_i$ is moving down the edge $(r, u)$ by the same distance. While $s_i$ is in the interior of $(r, u)$, no server $q$ from some other branch is adjacent to $v$, because the unique path between $v$ and $q$ passes through $s_i$.

lower bound for the WFA (Theorem 1.2) for depth-3 HSTs. The extension of Theorem 1.2 to the line is discussed in Appendix B.

## 2 LOWER BOUND FOR THE DC ALGORITHM ON DEPTH-2 HSTS

We now show a lower bound of $\Omega(h)$ on the competitive ratio of the DC algorithm.

Let $T$ be a depth-2 HST with $k + 1$ subtrees and edge lengths chosen as follows. Edges from the root $r$ to its children have length $1 - \epsilon$, and edges from the leaves to their parents length $\epsilon$ for some $\epsilon \ll 1$. Let $T_u$ be a subtree rooted at an internal node $u \neq r$. A *branch* $B_u$ is defined as $T_u$ together with the edge $e$ connecting $T_u$ to the root. We call $B_u$ *empty* if there is no online server in $T_u$ nor in the interior of $e$. Since $T$ contains $k + 1$ branches, at least one of them is always empty.

The idea behind the lower bound is quite simple. The adversary moves all its $h$ servers to the leaves of an empty branch $B_u$, and keeps requesting those leaves until DC brings $h$ servers to $T_u$. Then, another branch has to become empty, and the adversary moves all its servers there, starting a new *phase*. The adversary can execute an arbitrary number of such phases.

The key observation is that DC is "too slow" when bringing new servers to $T_u$, and incurs a cost of order $\Omega(h^2)$ during each phase, while the adversary only pays $O(h)$.

Theorem 1.1. *The competitive ratio of* DC *in depth-2 HSTs is* $\Omega(h)$*, even when* $k/h \to \infty$.

Proof. We describe a phase, which can be repeated arbitrarily many times. The adversary places all its $h$ servers at different leaves of an empty branch $B_u$ and does not move until the end of the phase. At each time during the phase, a request arrives at such a leaf, which is occupied by some offline server, but contains no online servers. The phase ends at the moment when the $h$th server of DC arrives to $T_u$.

Let ALG denote the cost of the DC algorithm and ADV the cost of the adversary during the phase. Clearly, ADV $= 2h$ in each phase: The adversary moves its $h$ servers to $T_u$ and does not incur any additional cost until the end of the phase. However, we claim that ALG $= \Omega(h^2)$, no matter where exactly the DC servers are located when the phase starts. To see that, let us call Step $i$ the part of the phase when DC has exactly $i - 1$ servers in $T_u$. Clearly, Step 1 consists of only a single request, which causes DC to bring one server to the requested leaf. So, the cost of DC for Step 1 is at least 1. To bound the cost in the subsequent steps, we make the following observation. □

Observation 2.1. *At the moment when a new server $s$ enters the subtree $T_u$, no other DC servers are located along the edge $e = (r, u)$.*

This follows from the construction of DC, which moves only servers adjacent to the request. At the moment when $s$ enters the edge $e$, no other server above $s$ can be inside $e$; see Figure 1.

We now focus on Step $i$ for $2 \leq i \leq h$. There are already $i - 1$ servers in $T_u$, and let $s_i$ be the next one, which is to arrive to $T_u$.

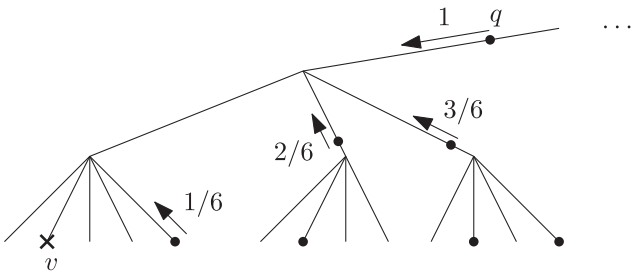


Fig. 2. A request at $v$, and Phase 2 of Algorithm 1. Note that $k_q^-$ equals 6 in the visualised case. Speed is noted next to each server moving.

Crucially, $s_i$ moves if and only if all the servers of DC in the subtree $T_u$ move from the leaves towards $u$, like in Figure 1: When the request arrives at $v$, $s_i$ moves by $\epsilon$ and the servers inside $T_u$ pay together $(i-1)\epsilon$. However, such a moment does not occur, until all servers $s_1, \ldots, s_{i-1}$ are again at leaves, i.e., they incur an additional cost $(i-1)\epsilon$. To sum up, while $s_i$ moves by $\epsilon$, the servers inside $T_u$ incur cost $2(i-1)\epsilon$.

When Step $i$ starts, the distance of $s_i$ from $u$ is at least $1-\epsilon$, and therefore $s_i$ moves by distance $\epsilon$ at least $\lfloor \frac{1-\epsilon}{\epsilon} \rfloor$ times, before it enters $T_u$. So, during Step $i$, DC pays at least

$$\left\lfloor \frac{1-\epsilon}{\epsilon} \right\rfloor (2(i-1)\epsilon + \epsilon) \geq \frac{1-2\epsilon}{\epsilon} \cdot \epsilon(2(i-1)+1) = (1-2\epsilon)(2i-1).$$

By summing over all steps $i = 1, \ldots, h$ and choosing $\epsilon \leq 1/4$, we get

$$\text{ALG} \geq 1 + \sum_{i=2}^{h} (1-2\epsilon)(2i-1) \geq \sum_{i=1}^{h} (1-2\epsilon)(2i-1) = (1-2\epsilon)h^2 \geq \frac{h^2}{2}.$$

To conclude the proof, we note that $\text{ALG}/\text{ ADV} \geq (h^2/2)/(2h) = h/4 = \Omega(h)$ for all phases. □

## 3 ALGORITHM FOR DEPTH-$d$ TREES

In this section, we prove Theorem 1.3.

Recall that a depth-$d$ tree is a rooted tree, and we allow the requests to appear only at the leaves. However, to simplify the algorithm description, we will allow the online servers to reside at any node or at any location on an edge (similar to that in DC). To serve a request at a leaf $v$, the algorithm moves all the adjacent servers towards $v$, where the speed of each server coming from a different branch of the tree depends on the number of the online servers "behind" it.

To describe the algorithm formally, we state the following definitions. Let $T$ be a depth-$d$ tree. For a point $x \in T$ (either a node or a location on some edge), we define $T_x$ as the subtree consisting of all points below $x$ including $x$ itself, and we denote $k_x$ the number of the online servers inside $T_x$. If $s$ is a server located at a point $x$, then we denote $T_s = T_x$ and $k_s = k_x$. We also denote $T_x^- = T_x \setminus \{x\}$, and $k_x^-$ the corresponding number of the algorithm's servers in $T_x^-$.

### 3.1 Algorithm Description

Suppose that a request arrives at a leaf $v$; let $A$ be the set of algorithm's servers adjacent to $v$. The algorithm proceeds in two phases, depending on whether there is a server along the path from $v$ to the root $r$ or not. We set speeds as described in Algorithm 1 below and move the servers towards $v$ either until the phase ends or the set $A$ changes. This defines the elementary moves (where $A$ stays unchanged). Note that if there are some servers in the path between $v$ and the root $r$, only the lowest of them belongs to $A$. Figure 2 shows the progress of Phase 2.

**ALGORITHM 1:** Serving request at leaf $v$.

**Phase 1:** While there is no server along the path $r - v$
For each $s \in A$: move $s$ at speed $k_s/k$

**Phase 2:** While no server reached $v$; Server $q \in A$ moves down along the path $r - v$
For server $q$: move it at speed 1
For each $s \in A \setminus \{q\}$: move it at speed $k_s/k_q^-$

We note two properties, the first of which follows directly from the design of Algorithm 1.

Observation 3.1. *No edge $e \in T$ contains more than one server of Algorithm 1 in its interior.*

Note that during both phases, the various servers move at non-uniform speeds, depending on their respective $k_s$. The following observations about these speeds will be useful.

Observation 3.2. *During Phase 1, the total speed of servers is 1. This follows as $\sum_{s\in A} k_s = k$. Analogously, during Phase 2, the total speed of servers inside $T_q^-$ is 1, if there are any. This follows as $\sum_{s\in A\setminus\{q\}} k_s = k_q^-$.*

The intuition behind the algorithm is the following. Recall that the problem with DC is that it moves its servers too slowly towards an active region when requests start arriving there. In contrast, we change the speeds of the servers adjacent to $v$ to make the algorithm more aggressive. From each region, an adjacent server moves at speed proportional to the number of the online servers in that region. This way, if a region has many servers and not many requests appear there (we call such regions excessive), servers move quickly from there to a more active region. Moreover, in Phase 2, server $q$ is viewed as a *helper* coming to aid the servers inside $T_q^-$. The second main idea is to keep the total speed of the helper proportional to the total speed of the servers inside $T_q^-$. This prevents the algorithm from becoming overly aggressive and keeps the cost of the moving helpers comparable to the cost incurred within $T_q^-$.

### 3.2 Analysis

We will analyze the algorithm based on a suitable potential function $\Phi(t)$. Let $\mathrm{ALG}(t)$ and $\mathrm{OPT}(t)$ denote the cost of the algorithm and of the adversary, respectively, for serving the request at time $t$. Let $\Delta_t\Phi = \Phi(t) - \Phi(t-1)$ denote the change of the potential at time $t$. We will ensure that $\Phi$ is non-negative and bounded from above by a function of $h, k, d$, and length of the longest edge in $T$. Therefore, to show $R$-competitiveness, it suffices to show that the following holds at each time $t$: $\mathrm{ALG}(t) + \Delta_t\Phi \le R \cdot \mathrm{OPT}(t)$.

To show this, we split the analysis into two parts: First, we let the adversary move a server (if necessary) to serve the request. Then, we consider the move of the algorithm. Let $\Delta_t^{OPT}\Phi$ and $\Delta_t^{ALG}\Phi$ denote the changes in $\Phi$ due to the move of the adversary and the algorithm, respectively. Clearly, $\Delta_t\Phi = \Delta_t^{OPT}\Phi + \Delta_t^{ALG}\Phi$, and thus it suffices to show the following two inequalities:

$$\Delta_t^{OPT}\Phi \le R \cdot OPT(t), \tag{1}$$

$$ALG(t) + \Delta_t^{ALG}\Phi \le 0. \tag{2}$$

*3.2.1 Potential Function.* Before we define the potential, we need to formalize the notion of excess and deficiency in the subtrees of $T$. Let $d(a, b)$ denote the distance of points $a$ and $b$. For $e = (u, v) \in T$, where $v$ is the node closer to the root, we define $k_e := k_u + \frac{1}{d(u,v)} \sum_{s\in e} d(s, v)$. Note that this is the number of online servers in $T_u$, plus the possible single server in $e$ counted fractionally, proportionally to its position along $e$. For an edge $e$, let $\ell(e)$ denote its level with the convention

that the edges from leaf to their parents have level 1, and the edges from root to its children have level $d$. For $\ell = 1, \ldots, d$, let $\beta_\ell$ be some geometrically increasing constants that will be defined later. For any point $x \in T$, similarly to $k_x$ and $k_x^-$, we denote by $h_x$ and $h_x^-$ the number of servers of the adversary in $T_x$ and $T_x^-$, respectively. For an edge $e$, we define the *excess* $E_e$ *of* $e$ and the *deficiency* $D_e$ *of* $e$ as follows:

$$E_e = \max\{k_e - \lfloor \beta_{\ell(e)} \cdot h_u \rfloor, 0\} \cdot d(u,v) \qquad D_e = \max\{\lfloor \beta_{\ell(e)} \cdot h_u \rfloor - k_e, 0\} \cdot d(u,v).$$

Note that these compare $k_e$ to $h_u$ with respect to the *excess threshold* $\beta_{\ell(e)}$. We call an edge *excessive*, if $E_e > 0$; otherwise, we call it *deficient*. Let us state a few basic properties of these two terms.

Observation 3.3. *Let $e$ be an edge containing an algorithm's server $s$ in its interior. If $e$ is excessive, then it cannot become deficient unless $s$ moves upwards completely outside of the interior of $e$. Similarly, if $e$ is deficient, then it cannot become excessive unless $s$ leaves interior of $e$ completely.*

Note that no other server can pass through $e$ while $s$ still resides there and the contribution of $s$ to $k_e$ is a nonzero value strictly smaller than 1, while $\lfloor \beta_\ell h_u \rfloor$ is an integer.

Observation 3.4. *Let $e$ be an edge and $s$ be an algorithm's server in its interior moving by a distance $x$. Then either $D_e$ or $E_e$ changes exactly by $x$.*

This is because $k_e$ changes by $x/d(u,v)$, and therefore the change of $D_e$ (respectively, $E_e$) is $x$.

Observation 3.5. *If an adversary server passes through the edge $e$, then change in $D_e$ (respectively, $E_e$) will be at most $\lceil \beta_{\ell(e)} \rceil \cdot d(u,v)$.*

To see this, note that $\lfloor \beta_{\ell(e)} h_u \rfloor \le \lfloor \beta_{\ell(e)}(h_u - 1) \rfloor + \lceil \beta_{\ell(e)} \rceil$.

We now fix the excess thresholds. We first define $\beta$ depending on $\delta = k/h$ as

$$\beta = 2 \text{ if } \delta \ge 2^d, \text{ and } \beta = \delta^{1/d} \text{ for } \delta \le 2^d.$$

For convenience in the calculations, we denote $\gamma := \frac{\beta}{\beta - 1}$. Note that, for all possible $\delta > 1$, our choices satisfy $1 < \beta \le 2$, $\gamma \ge 2$, and

$$\beta \le \delta^{1/d}. \tag{3}$$

For each $\ell = 1, \ldots, d$, we define the excess threshold for all edges in the level $\ell$ as $\beta_\ell := \beta^{\ell - 1}$.

Now, we can define the potential. Let

$$\Phi := \sum_{e \in T} \left( \alpha^D_{\ell(e)} D_e + \alpha^E_{\ell(e)} E_e \right),$$

where the coefficients $\alpha^D_\ell$ and $\alpha^E_\ell$ are as follows:

$$\alpha^D_\ell := 2\ell - 1, \qquad \text{for } \ell = 1, \ldots, d,$$

$$\alpha^E_d := \gamma \left( 1 + \frac{1}{\beta} \alpha^D_d \right),$$

$$\alpha^E_\ell := \sum_{i=\ell}^{d-1} \gamma^{i-\ell+1} \left( 2 + \frac{1}{\beta} \alpha^D_i \right) + \gamma^{d-\ell} \alpha^E_d \qquad \text{for } \ell = 1, \ldots, d-1.$$

Note that $\alpha^D_\ell > \alpha^D_{\ell-1}$ and $\alpha^E_\ell < \alpha^E_{\ell-1}$ for all $1 < \ell \le d$. The latter follows as the multipliers $\gamma^{i-\ell+1}$ and $\gamma^{d-\ell}$ decrease with increasing $\ell$ and moreover the summation in the first term of $\alpha^E_\ell$ has fewer terms as $\ell$ increases.

To prove the desired competitive ratio for Algorithm 1, the idea will be to show that the *good* moves (when a server enters a region with deficiency, or leaves a region with excess) contribute

more than the *bad* moves (when a server enters a region with excess or leaves a region that is already deficient).

As the dynamics of the servers can be decomposed into elementary moves, it suffices to only analyze these. We will assume that no servers of $A$ are located at a node. This is without loss of generality, as only the moving servers can cause a change in the potential, and each server appears at a node just for an infinitesimal moment during its motion. Note that this assumption implies that each edge $e$ containing a server $s \in A$ is either excessive or deficient, i.e., either $E_e > 0$ or $D_e > 0$.

The following two lemmas give some properties of the deficient and excessive subtrees, which will be used later in the proof of Theorem 1.3.

Lemma 3.6 (Excess in Phase 1). *Consider a moment during Phase 1 and assume that no server of $A$ resides at a node. For the set $E = \{s \in A \mid s \in e, E_e > 0\}$ of servers that are located in excessive edges, and for $D = A \setminus E$, the following holds:*

$$\sum_{s \in D} k_s \leq \frac{1}{\beta} k \qquad \text{and} \qquad \sum_{s \in E} k_s \geq \frac{1}{\gamma} k.$$

Proof. During Phase 1, each server of the algorithm resides in $T_s$ for some $s \in E \cup D$; therefore $k = \sum_{s \in E} k_s + \sum_{s \in D} k_s$. For each $s \in D$, let $\ell(s)$ be the level of the edge $e$ containing $s$. We have that $k_s \leq \lfloor \beta_{\ell(s)} \cdot h_s \rfloor$, otherwise $E_e$ would be positive. Therefore, we have

$$\sum_{s \in D} k_s \leq \sum_{s \in D} \lfloor \beta_{\ell(s)} h_s \rfloor \leq \sum_{s \in D} \beta_d \cdot h_s \leq \beta_d \cdot h = \beta^{d-1} \frac{k}{\delta} \leq \beta^{d-1} \frac{k}{\beta^d} = \frac{1}{\beta} k,$$

where the last inequality comes from Equation (3).

To prove the second inequality, observe that

$$\sum_{s \in E} k_s = k - \sum_{s \in D} k_s \geq k - \frac{1}{\beta} k = \left(1 - \frac{1}{\beta}\right) k = \frac{1}{\gamma} k. \qquad \square$$

Lemma 3.7 (Excess in Phase 2). *Consider a moment during Phase 2 and assume that no server of $A$ resides at a node. Let $\ell$ be the level of the edge containing $q$ and $A' = A \setminus \{q\}$. For the set $E = \{s \in A' \mid s \in e, E_e > 0\}$ of servers that are located in excessive edges, and for $D = A' \setminus E$, the following holds. If $k_q^- \geq \lfloor \beta_\ell h_q^- \rfloor$, then we have*

$$\sum_{s \in D} k_s \leq \frac{1}{\beta} k_q^- \qquad \text{and} \qquad \sum_{s \in E} k_s \geq \frac{1}{\gamma} k_q^-.$$

The proof of this lemma is quite similar to the previous one and it is deferred to Appendix A.1.

*3.2.2 Proof of Theorem 1.3.* We now show the main technical result, which directly implies Theorem 1.3.

Theorem 3.8. *The competitive ratio of Algorithm 1 in depth-$d$ trees is $O(d \cdot \gamma^d)$.*

This implies Theorem 1.3 as follows. If $\delta \geq 2^d$, then we have $\beta = 2$ and $\gamma = 2$, and we get the competitive ratio $O(d \cdot 2^d)$. For $1 < \delta < 2^d$, we have $\beta = \delta^{1/d}$ and therefore the competitive ratio is $O(d \cdot (\frac{\delta^{1/d}}{\delta^{1/d}-1})^d)$. In particular, if $\delta = (1+\epsilon)$ for some $0 < \epsilon \leq 1$, then we have $\beta = (1+\epsilon)^{1/d} \geq 1 + \frac{\epsilon}{2d}$. This implies that $\gamma = \frac{\beta}{\beta-1} \leq \frac{\beta}{\epsilon/(2d)} = \frac{\beta 2d}{\epsilon}$. Using Equation (3), we get that

$$\gamma^d \leq \beta^d \cdot \left(\frac{2d}{\epsilon}\right)^d \leq \delta \cdot \left(\frac{2d}{\epsilon}\right)^d = O\left(\left(\frac{2d}{\epsilon}\right)^d\right).$$

Thus, we get the ratio $O(d \cdot (\frac{2d}{\epsilon})^d)$.

We now prove Theorem 3.8.

Proof of Theorem 3.8. As $\Phi$ is non-negative and bounded from above by a function of $h, k, d$, and the length of the longest edge in $T$, it suffices to show the inequalities Equations (1) and (2).

We start with Equation (1), which is straightforward. By Observation 3.5, the move of a single adversary's server through an edge $e$ of length $x_e$ changes $D_e$ or $E_e$ in the potential by at most $\lceil \beta_{\ell(e)} \rceil x_e$. As the adversary incurs cost $x_e$ during this move, we need to show the following inequalities:

$$\lceil \beta_\ell \rceil x_e \cdot \alpha_\ell^D \le R \cdot x_e, \quad \text{for all } 1 \le \ell \le d,$$
$$\lceil \beta_\ell \rceil x_e \cdot \alpha_\ell^E \le R \cdot x_e, \quad \text{for all } 1 \le \ell \le d.$$

As we show in Lemma 3.11 below, $\lceil \beta_\ell \rceil \alpha_\ell^D$ and $\lceil \beta_\ell \rceil \alpha_\ell^E$ are of order $O(d \cdot \gamma^d)$. Therefore, setting $R = \Theta(d \cdot \gamma^d)$ will satisfy Equation (1).

We now consider Equation (2), which is much more challenging to show. Let us denote $A_E$ the set of edges containing some server from $A$ in their interior. We call an *elementary step* a part of the motion of the algorithm during which $A$ and $A_E$ remain unchanged, and all the servers of $A$ are located in the interior of the edges of $T$. Lemmas 3.9 and 3.10 below show that Equation (2) holds during an elementary step, and the theorem would follow by summing Equation (2) over all the elementary steps. □

Lemma 3.9. *During an elementary step in Phase 1 of Algorithm 1, the inequality Equation (2) holds.*

Proof. Without loss of generality, let us assume that the elementary step lasted exactly 1 unit of time. This makes the distance traveled by each server equals to its speed, and makes calculations cleaner.

Let ALG denote the cost incurred by the algorithm during this step. Note that ALG = 1, since by Observation 3.2, the total speed of the servers in $A$, is 1.

To estimate the change of the potential $\Delta\Phi$, we decompose $A$ into two sets, called $D$ and $E$. $D$ is the set of the servers of $A$ residing in deficient edges, and $E$ are the servers residing in excessive edges. Next, we evaluate $\Delta\Phi$ due to the movement of the servers from each class separately.

The movement of servers of $E$ is *good*, i.e., decreases the excess in their edges, while the movement of servers of $D$ increases the deficiency. By taking the largest possible $\alpha^D$ (which is $\alpha_d^D$) and the smallest possible $\alpha^E$ (which is $\alpha_d^E$) coefficient in the estimation, we can bound the change of the potential due to the move of the servers of $A$ as

$$\Delta\Phi \le \alpha_d^D \sum_{s \in D} \frac{k_s}{k} - \alpha_d^E \sum_{s \in E} \frac{k_s}{k} \le \frac{1}{\beta}\alpha_d^D - \frac{1}{\gamma}\alpha_d^E,$$

where the last inequality holds due to the Lemma 3.6. We get that

$$\text{ALG} + \Delta\Phi \le 1 + \frac{1}{\beta}\alpha_d^D - \frac{1}{\gamma}\alpha_d^E = 1 + \frac{1}{\beta}\alpha_d^D - \frac{1}{\gamma} \cdot \gamma \left(1 + \frac{1}{\beta}\alpha_d^D\right) = 0.$$ □

Lemma 3.10. *During an elementary step in Phase 2 of Algorithm 1, the inequality Equation (2) holds.*

Proof. Similarly to the proof of the preceding lemma, we denote by ALG the cost incurred by the algorithm and we assume (without loss of generality) that the duration of the elementary step is exactly 1 time unit, so that the speed of each server equals the distance it travels. As the speed of the server $q$ is 1, it also moves by distance 1. We denote by $\ell$ the level of the edge containing $q$.

We first consider the case $\ell = 1$. Here, $q$ is the only server that moves, thus ALG = 1. Let $e = (u, v)$ be the edge containing $q$, where $v$ is the requested leaf. The adversary has already a server at $v$, which implies that $e$ is deficient. Thus, the movement of $q$ decreases the deficiency of $e$. Since $\alpha_1^D = 1$, we have that ALG + $\Delta\Phi = 1 - 1 = 0$ and Equation (2) holds.

We now focus on the case $\ell > 1$. By Observation 3.2, the servers in $T_q^-$ (if any) move in total by $\sum_{s\in A\setminus\{q\}} \frac{k_s}{k_q^-} = 1$, and therefore ALG $\leq 2$. To estimate the change of the potential, we consider two cases.

(1) *When* $k_q^- \geq \lfloor \beta_\ell h_q^- \rfloor$. Here the movement of $q$ increases the excess in the edge containing $q$. Let us denote $E$ (respectively, $D$) the servers of $A \setminus \{q\}$ residing in the excessive (respectively, deficient) edges. By taking the largest possible $\alpha^D$ (which is $\alpha_{\ell-1}^D$) and the smallest possible $\alpha^E$ (which is $\alpha_{\ell-1}^E$) coefficient in the estimation, we can upper bound the change of the potential due to the move of the servers in $T_q^-$ as

$$\alpha_{\ell-1}^D \sum_{s\in D} \frac{k_s}{k_q^-} - \alpha_{\ell-1}^E \sum_{s\in E} \frac{k_s}{k_q^-} \leq \frac{1}{\beta}\alpha_{\ell-1}^D - \frac{1}{\gamma}\alpha_{\ell-1}^E,$$

where the above inequality holds due to the Lemma 3.7. As the movement of $q$ itself causes an increase of $\Phi$ by $\alpha_\ell^E$, we have

$$ALG + \Delta\Phi \leq 2 + \frac{1}{\beta}\alpha_{\ell-1}^D - \frac{1}{\gamma}\alpha_{\ell-1}^E + \alpha_\ell^E.$$

To see that this is non-positive, recall that $\alpha_\ell^E = \sum_{i=\ell}^{d-1} \gamma^{i-\ell+1}(2 + \frac{1}{\beta}\alpha_i^D) + \gamma^{d-\ell}\alpha_d^E$. Therefore,

$$\begin{aligned}
\frac{1}{\gamma}\alpha_{\ell-1}^E &= \frac{1}{\gamma}\sum_{i=\ell-1}^{d-1} \gamma^{i-\ell+2}\left(2 + \frac{1}{\beta}\alpha_i^D\right) + \frac{1}{\gamma}\gamma^{d-\ell+1}\alpha_d^E \\
&= \left(2 + \frac{1}{\beta}\alpha_{\ell-1}^D\right) + \sum_{i=\ell}^{d-1} \gamma^{i-\ell+1}\left(2 + \frac{1}{\beta}\alpha_i^D\right) + \gamma^{d-\ell}\alpha_d^E \\
&= 2 + \frac{1}{\beta}\alpha_{\ell-1}^D + \alpha_\ell^E.
\end{aligned}$$

(2) *When* $k_q^- < \lfloor \beta_\ell h_q^- \rfloor$. This case is much simpler. All the movement inside of $T_q^-$ might contribute to the increase of deficiency at level at most $\ell - 1$. However, $q$ then causes a decrease of deficiency at level $\ell$, and we have ALG + $\Delta\Phi \leq 2 + \alpha_{\ell-1}^D - \alpha_\ell^D$. This is less or equal to 0, as $\alpha_\ell^D \geq \alpha_{\ell-1}^D + 2$. □

Lemma 3.11. *For each* $1 \leq \ell \leq d$, *both* $\lceil \beta_\ell \rceil \alpha_\ell^D$ *and* $\lceil \beta_\ell \rceil \alpha_\ell^E$ *are of order* $O(d \cdot \gamma^d)$.

Proof. We have defined $\alpha_\ell^D = 2\ell - 1$, and therefore

$$\lceil \beta_\ell \rceil \alpha_\ell^D \leq 2 \cdot \beta^\ell (2\ell - 1) \leq 2 \cdot \beta^d (2d - 1) = O(d \cdot \gamma^d),$$

since we have chosen $1 < \beta \leq 2$ and $\gamma \geq 2$ for any possible $\delta$.

Now, we focus on $\alpha_\ell^E$. Note that $\alpha_d^E = \gamma(1 + \frac{1}{\beta}\alpha_d^D)$, which is $O(d \cdot \gamma)$, as $\beta > 1$ and $\alpha_d^D = O(d)$. For $\alpha_\ell^E$, we have

$$\begin{aligned}\alpha_\ell^E &= \sum_{i=\ell}^{d-1} \gamma^{i-\ell+1}\left(2 + \frac{1}{\beta}\alpha_i^D\right) + \gamma^{d-\ell}\alpha_d^E \le \left(2 + \frac{1}{\beta}\alpha_{d-1}^D\right)\sum_{i=\ell}^{d-1}\gamma^{i-\ell+1} + \gamma^{d-\ell}\alpha_d^E \\ &= \left(2 + \frac{1}{\beta}\alpha_{d-1}^D\right)\sum_{j=1}^{d-\ell}\gamma^j + \gamma^{d-\ell}\alpha_d^E = \left(2 + \frac{1}{\beta}\alpha_{d-1}^D\right)\frac{\gamma\cdot(\gamma^{d-\ell}-1)}{\gamma-1} + \gamma^{d-\ell}\alpha_d^E \\ &\le \beta\left(2 + \frac{1}{\beta}\alpha_{d-1}^D\right)\gamma^{d-\ell} + \gamma^{d-\ell}\alpha_d^E = \gamma^{d-\ell}(2\beta + 2d - 3 + \alpha_d^E) = O(\gamma^{d-\ell+1}\cdot d),\end{aligned}$$

where we used that $\frac{\gamma}{\gamma-1} = \beta$, $1 < \beta \le 2$ and $\alpha_d^E = O(\gamma \cdot d)$. Therefore, as $\beta_\ell \le \gamma^{\ell-1}$, we have $\lceil\beta_\ell\rceil\alpha_\ell^E = O(\gamma^d d)$, and this concludes the proof. □

## 4 LOWER BOUNDS

In this section, we prove Theorems 1.4 and 1.2. We first show a general lower bound on the competitive ratio of any algorithm for depth-2 HSTs. Then, we give lower bound on the competitive ratio of WFA. Similarly to the previous section, given a tree $T$ and a point $x \in T$ (either a node or a location on some edge), we define $T_x$ as the subtree consisting of all points below $x$ including $x$ itself. If $u$ is a parent of a leaf, then we call $T_u$ an *elementary subtree*.

### 4.1 General Lower Bound for Depth-2 HSTs

We now give a lower bound on the competitive ratio of any deterministic online algorithm on depth-2 HSTs. In particular, we show that for sufficiently large $h$, any deterministic online algorithm has competitive ratio at least 2.4.

The metric space is a depth-2 HST $T$ with the following properties: $T$ contains at least $k+1$ elementary subtrees and each one of them has at least $h$ leaves. To ease our calculations, we assume that edges of the lower level have length $\epsilon \ll 1$ and edges of the upper level have length $1-\epsilon$. So, the distance between leaves of different elementary subtrees is 2.

Theorem 1.4 (Restated). *For sufficiently large $h$, even when $k/h \to \infty$, there is no 2.4-competitive deterministic online algorithm, even for depth-2 HSTs.*

Proof. Without loss of generality both the online and offline algorithms are "lazy," i.e., they move a server only for serving a request (this is a folklore $k$-server property; see, e.g., Borodin and El-Yaniv (1998)). Since requests only appear at leaves of $T$, all the offline and online servers are always located at the leaves. We say that a server is inside an elementary subtree $T_u$, if it is located at some leaf of $T_u$. If there are no online servers at the leaves of $T_u$, then we say that $T_u$ is *empty*. Observe that at any given time there exists at least one empty elementary subtree.

Let $\mathcal{A}$ be an online algorithm. The adversarial strategy consists of arbitrarily many iterations of a phase. During a phase, some offline servers are moved to an empty elementary subtree $T_u$ and requests are made there until the cost incurred by $\mathcal{A}$ is sufficiently large. At this point the phase ends and a new phase may start in another empty subtree. Let ALG and ADV denote the cost of $\mathcal{A}$ and the adversary, respectively, during a phase. We will ensure that for all phases ALG $\ge$ 2.4 $\cdot$ ADV. This implies the lower bound on the competitive ratio of $\mathcal{A}$.

We describe a phase of the adversarial strategy. The adversary moves some $\ell \le h$ servers to the empty elementary subtree $T_u$ and makes requests at leaves of $T_u$ until $\mathcal{A}$ brings $m$ servers there. In particular, each request appears at a leaf of $T_u$ that is not occupied by a server of $\mathcal{A}$. We denote by

$s(i)$ the cost that $\mathcal{A}$ has to incur for serving requests inside $T_u$ until it moves its $i$th server there (this does not include the cost of moving the server from outside $T_u$). Clearly, $s(1) = 0$ and $s(i) \leq s(i+1)$ for all $i > 0$. The choice of $\ell$ and $m$ depends on the values $s(i)$ for $2 \leq i \leq h$. We will now show that for any values of $s(i)$'s, the adversary can choose $\ell$ and $m$ such that ALG $\geq 2.4 \cdot$ ADV.

First, if there exists an $i$ such that $s(i) \geq 3i$, we set $\ell = m = i$. Intuitively, the algorithm is too slow in bringing its servers to $T_u$ in this case. Both $\mathcal{A}$ and the adversary incur a cost of $2i$ to move $i$ servers to $T_u$. However, $\mathcal{A}$ pays a cost of $s(i)$ for serving requests inside $T_u$, while the adversary can serve all those requests at zero cost (all requests can be located at leaves occupied by offline servers). Overall, the cost of $\mathcal{A}$ is $2i + s(i) \geq 5i$, while the offline cost is $2i$. Thus, we get that ALG $\geq 2.5 \cdot$ ADV.

Similarly, if $s(i) \leq (10i - 24)/7$ for some $i$, we choose $\ell = 1$ and $m = i$. Roughly speaking, in that case the algorithm is too "aggressive" in bringing its first $i$ servers, thus incurring a large movement cost. Here, the adversary only moves one server to $T_u$. Each request is issued at an empty leaf of $T_u$. Therefore $\mathcal{A}$ pays for each request in $T_u$ and the same holds for the single server of the adversary. So, ALG $= 2i + s(i)$ and ADV $= 2 + s(i)$. By our assumption on $s(i)$, this gives

$$\text{ALG} - 2.4 \cdot \text{ADV} = 2i + s(i) - 4.8 - 2.4 \cdot s(i) = 2i - 4.8 - 1.4 \cdot s(i) \geq 0.$$

We can thus restrict our attention to the case that $s(i) \in (\frac{10i-24}{7}, 3i)$, for all $2 \leq i \leq h$. Now, we want to upper bound the offline cost, for $1 < \ell < h$ and $m = h$. Clearly, the cost of moving $\ell$ offline servers into $T_u$ is $2\ell$, and for the time that $\mathcal{A}$ has less than $\ell$ servers in $T_u$, the cost of moving offline servers within $T_u$ is zero. It remains to count the offline cost during the time that $\mathcal{A}$ has at least $\ell$ servers inside $T_u$.

For the part of the request sequence when $\mathcal{A}$ has $\ell \leq j < h$ servers in $T_u$, it incurs a cost $s(j+1) - s(j)$. Since the problem restricted to an elementary subtree is equivalent to the paging problem, using the lower bound result of Sleator and Tarjan (1985) for paging,[2] we get that the adversary incurs a cost of at most $(s(j+1) - s(j)) \cdot \frac{j-\ell+1}{j} + 2\epsilon j \leq (s(j+1) - s(j)) \cdot \frac{h-\ell}{h-1} + 2\epsilon j$. Also, there are $h - \ell$ requests where $\mathcal{A}$ brings new servers. Clearly, those requests cost to the adversary at most $(h - \ell)2\epsilon$. Thus, the cost of the adversary inside $T_u$ is at most

$$\sum_{j=\ell}^{h-1} \left( (s(j+1) - s(j)) \cdot \frac{h-\ell}{h-1} + 2\epsilon j \right) + (h-\ell)2\epsilon = \frac{h-\ell}{h-1}(s(h) - s(\ell)) + 2\epsilon \sum_{j=\ell}^{h-1} j + (h-\ell)2\epsilon.$$

For any value of $h$, we can take $\epsilon$ small enough, such that the terms involving $\epsilon$ tend to zero. Thus, the upper bound on the offline cost is arbitrarily close to $2\ell + (s(h) - s(\ell)) \cdot \frac{h-\ell}{h-1}$, where the first term is the cost of moving $\ell$ servers to $T_u$. Let us denote $c = s(h)/h$. Note that assuming $h$ is large enough, $c \in (1, 3)$. We get that

$$\frac{\text{ALG}}{\text{ADV}} \geq \frac{2h + s(h)}{2\ell + (s(h) - s(\ell)) \cdot \frac{h-\ell}{h-1}} \geq \frac{2h + ch}{2\ell + (ch - \frac{10\ell-24}{7}) \cdot \frac{h-\ell}{h-1}} = \frac{2h + ch}{2\ell + h \cdot (c - \frac{10\ell-24}{7h}) \cdot \frac{h-\ell}{h-1}}. \quad (4)$$

We now show that for every value of $c \in (1, 3)$, there is an $\ell = \lfloor \beta h \rfloor$, where $\beta \in (0, 1)$ is a constant depending on $c$, such that this ratio is at least 2.4. First, as $h$ is large enough, the right-hand side

[2] Sleator and Tarjan (1985) show that an adversary with $h$ servers can create a request sequence against an online algorithm with $k$ servers such that the cost of the adversary is at most $\frac{k-h+1}{k}$ ALG $+ (k - h + 1) \cdot D$, where $D$ is the distance between two leaves.

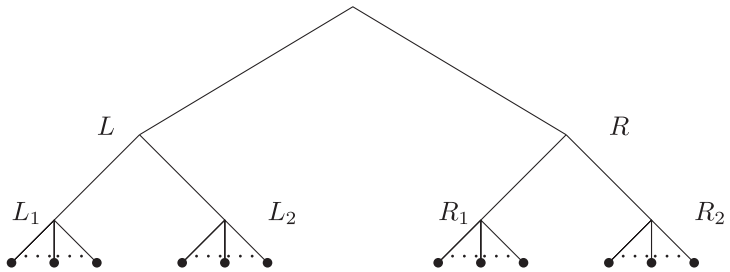


Fig. 3. The tree where the lower bound for WFA is applied: All requests arrive at leaves of $L_1, L_2, R_1$ and $R_2$.

of Equation (4) is arbitrarily close to

$$\frac{2h+ch}{2\ell+h\cdot(c-10\ell/7h)\cdot(1-\ell/h)} = \frac{2h+ch}{2\ell+h(c-c\ell/h-10\ell/(7h)+10\ell^2/(7h^2))}$$
$$\geq \frac{2h+ch}{2\beta h+h(c-c(\beta h-1)/h-10(\beta h-1)/(7h)+10\beta^2h^2/(7h^2))}$$
$$= \frac{2+c}{2\beta+c-c\beta+c/h-10\beta/7+10/(7h)+10\beta^2/7} \to \frac{2+c}{(10/7)\cdot\beta^2+(4/7-c)\beta+c}.$$

We choose $\beta := (c-4/7)/(20/7)$. Note that, as $c \in (1,3)$, we have that $\beta < 1$, and hence $\ell < h$. The expression above then evaluates to $(2+c)/(c-(c-\frac{4}{7})^2/\frac{40}{7})$. By standard calculus, this expression is minimized at $c = (2\sqrt{221}-14)/7$, where it attains a value higher than 2.419. □

### 4.2 Lower Bound for WFA on Depth-3 HSTs

We give an $\Omega(h)$ lower bound on the competitive ratio of the Work Function Algorithm (WFA) in the $(h,k)$-setting. More precisely, we show a lower bound of $h+1/3$ for $k=2h$. We first present the lower bound for a depth-3 HST. In Appendix B, we show how the construction can be adapted to work for the line. We also show that this exactly matches the upper bound of $(h+1)\cdot \mathrm{OPT}_h - \mathrm{OPT}_k$ implied by results of Bartal and Koutsoupias (2004) and Koutsoupias (1999) for the line.

The basic idea behind the lower bound is to trick the WFA to use only $h$ servers for servicing requests in an "active region" for a long time before it moves its extra available online servers. Moreover, we make sure that during the time the WFA uses $h$ servers in that region, it incurs an $\Omega(h)$ times higher cost than the adversary. Finally, when the WFA brings its extra servers, the adversary moves all its servers to some different region and starts making requests there. So, eventually, WFA is unable to use its additional servers in a useful way to improve its performance.

Theorem 1.2 (Restated). *The competitive ratio of the WFA in a depth-3 HST for $k=2h$ is at least $h+1/3$.*

Proof. Let $T$ be a depth-3 HST. We assume that the lengths of the edges in a root-to-leaf path are $\frac{1-\epsilon}{2}, \frac{\epsilon-\epsilon'}{2}, \frac{\epsilon'}{2}$, for $\epsilon' \ll \epsilon \ll 1$. So, the diameter of $T$ is 1 and the diameter of depth-2 subtrees is $\epsilon$. We also assume that $N = \frac{1}{\epsilon}$ is an integer such that $N \gg 3h^3$. Let $L$ and $R$ be two subtrees of depth 2. Inside each one of them, we focus on 2 elementary subtrees $L_1, L_2$ and $R_1, R_2$, respectively (see Figure 3), each one containing exactly $h$ leaves. All requests appear at leaves of those elementary subtrees. Initially, all servers are at leaves of $L$. Thus, both the WFA and the adversary always have their servers at leaves. This way, we say that some servers of the WFA (or the adversary) are inside a subtree, meaning that they are located at some leaves of that subtree.

The adversarial strategy consists of arbitrary many iterations of a phase. At the beginning of the phase, all online and offline servers are in the same subtree, either $L$ or $R$. At the end of the phase,

all servers have moved to the other subtree ($R$ or $L$, respectively), so a new phase may start. Before describing the phase, we give the basic strategy, which is repeated many times. For any elementary subtree $T$ with leaves $t_1, \ldots, t_h$, any $1 \leq \ell \leq h$ and any $c > 0$, we define strategy $S(T, \ell, c)$.

*Strategy $S(T, \ell, c)$:*

(1) While the WFA has $i < \ell$ servers in $T$: Request points $t_1, \ldots, t_\ell$ in an adversarial way (i.e., each time request a point where the WFA does not have a server).
(2) If $\ell \geq 2$ and the WFA has $i \geq \ell$ servers in $T$, then: While optimal cost to serve all requests using $\ell - 1$ servers (starting at $t_1, \ldots, t_{\ell-1}$) is smaller than $c$, request points $t_1, \ldots, t_\ell$ in a round-robin way.

Note that the second part might not be executed at all, depending on the values of $\ell$ and $c$.

We now describe a left-to-right phase, i.e., a phase that starts with all servers at $L$ and ends with all servers at $R$. A right-to-left phase is completely symmetric, i.e., we replace $R$ by $L$ and $R_i$ by $L_i$. Recall that $N = \frac{1}{\epsilon}$.

**Left-to-right phase:**

**Step 1:** For $\ell = 1$ to $h$: Apply strategy $S(R_1, \ell, 2)$

**Step 2:** For $i = 1$ to $N + 1$:
For $\ell = 1$ to $h$: Apply strategy $S(R_2, \ell, 2\epsilon)$
For $\ell = 1$ to $h$: Apply strategy $S(R_1, \ell, 2\epsilon)$

Intuitively, the WFA starts with no server at $R$ and at the end of Step 1, it has $h$ servers there (more precisely, in $R_1$). During Step 2, the WFA moves all its servers to $R$, as we show later.

Let ALG and ADV be the online and offline cost, respectively. We now state two basic lemmas for evaluating ALG and ADV during each step. Proofs of those lemmas, require a characterization of work function values, which comes later on. Here, we just give the intuition behind the proofs.

Lemma 4.1. *During Step 1,* $\text{ALG} \geq h^2 - \epsilon' \cdot (h-1)h$ *and* $\text{ADV} = h$.

Intuitively, we will exploit the fact that the WFA moves its servers too slowly towards $R$. In particular, we show (Lemma 4.21) that whenever the WFA has $i < h$ servers in $R$, it incurs a cost of at least $(2 - 2\epsilon') \cdot i$ inside $R$ before it moves its $(i+1)$th server there. This way, we get that the cost of the algorithm is at least $h + \sum_{i=1}^{h-1}(2 - 2\epsilon') \cdot i = h^2 - \epsilon' \cdot (h-1)h$. However, the adversary moves its $h$ servers by distance 1 (from $L$ to $R_1$) and then it serves all requests at zero cost.

Lemma 4.2. *During Step 2,* $\text{ALG} \geq 2h^2 + h - 3\epsilon h^3 - 2\frac{\epsilon'}{\epsilon} \cdot (h-1) \cdot h$ *and* $\text{ADV} = (2 + 2\epsilon)h$.

Roughly speaking, to prove this lemma, we make sure that for almost the whole Step 2, the WFA has $h$ servers in $R$ and they incur a cost $h$ times higher than the adversary. The additional servers move to $R$ almost at the end of the phase. The cost of the adversary is easy to calculate. There are $N + 1 = 1/\epsilon + 1$ iterations, where in each one of them the adversary moves its $h$ servers from $R_1$ to $R_2$ and then back to $R_1$, incurring a cost of $2\epsilon \cdot h$.

The theorem follows from Lemmata 4.1 and 4.2. Summing up for the whole phase, the offline cost is $(3 + 2\epsilon) \cdot h$ and the online cost is at least $3h^2 + h - 3\epsilon h^3 - \frac{\epsilon'}{\epsilon} 2(h-1)h - \epsilon'(h-1)h$. We get that

$$\frac{\text{ALG}}{\text{ADV}} \geq \frac{3h^2 + h - 3\epsilon h^3 - \frac{\epsilon'}{\epsilon} 2(h-1)h - \epsilon'(h-1)h}{(3 + 2\epsilon)h} \to \frac{3h^2 + h}{3h} = h + \frac{1}{3}. \qquad \square$$

*Work Function Values.* We now give a detailed characterization of how the work function evolves during left-to-right phases. For right-to-left phases the structure is completely symmetric.

**Basic Properties:** We state some standard properties of work functions that we use extensively in the rest of this section. Proofs of those properties can be found, e.g., in Borodin and El-Yaniv (1998). First, for any two configurations $X, Y$ at distance $d(X, Y)$ the work function $w$ satisfies

$$w(X) \leq w(Y) + d(X, Y).$$

Also, let $w$ and $w'$ be the work function before and after a request $r$, respectively. For a configuration $X$ such that $r \in X$, we have that $w'(X) = w(X)$. Last, let $C$ and $C'$ be the configurations of the WFA before and after serving $r$, respectively. The work function $w'$ satisfies $w'(C) - w'(C') = d(C, C')$.

**Notation:** Let $(L^i, R^{k-i})$ denote a configuration that has $i$ servers at $L$ and $(k - i)$ servers at $R$. Let $w(L^i, R^{k-i})$ be the minimum work function value of a configuration $(L^i, R^{k-i})$. If we need to make precise how many servers are in each elementary subtree, then we denote by $(L^i, r_1, r_2)$ a configuration that has $i$ servers at $L$, $r_1$ servers at $R_1$ and $r_2$ servers at $R_2$. Same as before, $w(L^i, r_1, r_2)$ denotes the minimum work function value of those configurations.

**Definition:** For two configurations $X$ and $Y$, we say that $X$ *supports* $Y$, if $w(Y) = w(X) + d(X, Y)$. The set of configurations that are not supported by any other configuration is called the *support* of work function $w$. The following observation will be useful later in our proofs.

**Initialization:** For convenience, we assume that at the beginning of the phase the minimum work function value of a configuration is zero. Note that this is without loss of generality: If $m = \min_X w(X)$ when the phase starts, then we just subtract $m$ from the work function values of all configurations. We require the invariant that at the beginning of the (left-to-right) phase, for $0 \leq i \leq k$:

$$w(L^{k-i}, R^i) = i.$$

This is clearly true for the beginning of the first phase, as all servers are initially at $L$. We are going to make sure, that the symmetric condition (i.e., $w(L^i, R^{k-i}) = m + i$) holds at the end of the phase, so a right-to-left phase may start.

We now state two auxiliary lemmata that would be helpful later.

Lemma 4.3. *Consider a left-to-right phase. At any time after the end of strategy $S(R_1, \ell, 2)$, for $1 \leq \ell \leq h$, we have that $w(L^{k-\ell+1}, R^{\ell-1}) = w(L^{k-\ell}, R^\ell) + 1$.*

At a high-level, the reason that this lemma holds is that any schedule $\mathcal{S}$ that uses no more than $\ell - 1$ servers in $R_1$, incurs a cost of at least 2 during the execution of $S(R_1, \ell, 2)$. Thus, by moving an $\ell$th server to $R_1$ (cost 1), and serving all requests of $S(R_1, \ell, 2)$ at zero cost, we obtain a schedule that is cheaper than $\mathcal{S}$ by at least 1. The formal proof is deferred to the end of this section.

Lemma 4.4. *Consider a left-to-right phase. At any time after the end of strategy $S(R_1, \ell, 2)$, for $1 \leq \ell \leq h$, we have that $w(L^{k-i}, R^i) = w(L^{k-\ell}, R^\ell) + (\ell - i)$ for any $0 < i < \ell$.*

Proof. We use backwards induction on $i$. Note that the base case $i = \ell - 1$ is true due to Lemma 4.3. Let $t$ be any time after the end of strategy $S(R_1, \ell, 2)$ with work function $w$.

*Induction Step:* Assume that the lemma is true for some $i < \ell$. We show that the lemma holds for $i - 1$. Clearly, time $t$ is after the end of $S(R_1, i, 2)$. From Lemma 4.3, we have that

$$w(L^{k-i+1}, R^{i-1}) = w(L^{k-i}, R^i) + 1. \tag{5}$$

From the inductive hypothesis, we have that

$$w(L^{k-i}, R^i) = w(L^{k-\ell}, R^\ell) + (\ell - i). \tag{6}$$

By Equations (5) and (6), we get that

$$w(L^{k-i+1}, R^{i-1}) = w(L^{k-i}, R^i) + 1 = w(L^{k-\ell}, R^\ell) + \ell - (i-1). \qquad \square$$

**First Step:** We now focus on the first step of the phase. Using the two lemmata above, we can characterize the structure of the work function at the end of the execution of $S(R_1, \ell, 2)$, for $1 \le \ell \le h$.

Lemma 4.5. *Consider a left-to-right phase. When strategy $S(R_1, \ell, 2)$ ends, for any $1 \le \ell \le h$, the following two things hold:*

*(1) For all $0 \le i < \ell$, $w(L^{k-i}, R^i) = w(L^{k-\ell}, R^\ell) + (\ell - i)$,*
*(2) For all $\ell < i \le k$, $w(L^{k-i}, R^i) = w(L^{k-\ell}, R^\ell) + (i - \ell)$.*

*In other words, having $\ell$ servers in $R$ is the lowest state, and all other states are tight with respect to it.*

Proof. For $i < \ell$, the lemma follows from Lemma 4.4. We focus on $i \ge \ell$: Until the end of $S(R_1, \ell, 2)$ there are $\ell \le i$ points in $R$ requested, so $w(L^{k-i}, R^i)$ does not change. Since at the beginning $w(L^{k-i}, R^i) = i$, we have that for all $i \ge \ell + 1$,

$$w(L^{k-i}, R^i) - w(L^{k-\ell}, R^\ell) = i - \ell. \qquad \square$$

This characterization of work function values is sufficient to give us the lower bound on the cost of the WFA during Step 1. The reader might prefer to move to Lemma 4.21, which estimates the cost of the WFA during the time interval that it has $i < h$ servers in $R$, and implies Lemma 4.1.

**Second step:** By Lemma 4.5, at the beginning of the second step, the work function values satisfy

$$w(L^{2h-i}, R^i) = w(L^h, R^h) + (h - i), \text{ for } 0 \le i < h, \tag{7}$$

$$w(L^{2h-i}, R^i) = w(L^h, R^h) + (i - h), \text{ for } h < i \le 2h. \tag{8}$$

We note that Equation (7) holds for the entire second step, due to Lemma 4.4.

Characterizing the structure of the work function during second step is more complicated. Note that Step 2 consists of $N + 1$ iterations, where in each iteration strategies $S(R_2, \ell, 2\epsilon)$ and $S(R_1, \ell, 2\epsilon)$ are applied. The following auxiliary lemma will be helpful in our arguments about the evolution of the work function during step 2.

Lemma 4.6. *For any $0 \le i < h$, the optimal schedule to serve $c$ consecutive iterations of Step 2 using $h + i$ servers in $R$, starting from a configuration $(L^{h-i}, h, i)$ and ending up in a configuration $(L^{h-i}, h - i', i + i')$, for $0 \le i' \le h - i$, has cost at least $c \cdot 2\epsilon(h - i) + \epsilon \cdot i'$.*

Proof. Consider a schedule $\mathcal{S}$ serving $c$ consecutive iterations of Step 2 using $h + i$ servers in $R$, starting from a configuration $(L^{h-i}, h, i)$. For $j = 1, \ldots, c$, let $(L^{h-i}, h - a_{j-1}, i + a_{j-1})$ be the configuration of $\mathcal{S}$ at the beginning of $j$th iteration. Let also $(L^{h-i}, i + b_j, h - b_j)$ be the configuration of $\mathcal{S}$ at the end of the execution of $S(R_2, h, 2\epsilon)$ during $j$th iteration. At the end of $j$th iteration, $\mathcal{S}$ is in configuration $(L^{h-i}, h - a_j, i + a_j)$ and the $(j+1)$th iteration starts. Note that $0 \le a_j, b_j \le h - i$, for all $1 \le j \le c$. Clearly, $a_0 = 0$, since we assume that $\mathcal{S}$ starts at a configuration $(L^{h-i}, h, i)$.

We now calculate the minimum cost of $\mathcal{S}$ depending on the values of $a_j$ and $b_j$. Consider the $j$th iteration. During executions of $S(R_2, \ell, 2\epsilon)$ there are $h - b_j - (i + a_{j-1}))$ servers that move from $R_1$ to $R_2$, incurring a movement cost of $\epsilon(h - b_j - i - a_{j-1})$. The cost of serving requests in $R_2$ is at least $2\epsilon \cdot b_j$, which is incurred during executions of $S(R_2, \ell, 2\epsilon)$ for $h - b_j < \ell \le h$. Similarly, the

movement cost for moving servers from $R_2$ to $R_1$ is $\epsilon(h - a_j - i - b_j)$ and the cost incurred inside $R_1$ is at least $2\epsilon a_j$. We get that the total cost of $\mathcal{S}$ is at least

$$\epsilon \cdot \sum_{j=1}^{c} (h - b_j - i - a_{j-1}) + (h - a_j - i - b_j) + 2a_j + 2b_j = \epsilon \cdot \sum_{j=1}^{c} 2h - 2i + a_j - a_{j-1}$$

$$= c \cdot 2\epsilon(h - i) + \epsilon \cdot \sum_{j=1}^{c} a_j - a_{j-1} = c \cdot 2\epsilon(h - i) + \epsilon \cdot a_c.$$

By setting $i' = a_c$, the lemma follows. □

Note that a possible schedule for serving an iteration using $h + i$ servers in $R$ starting from a configuration $(L^{h-i}, h, i)$ is to move $h - i$ servers from $R_1$ to $R_2$ (cost $\epsilon(h - i)$), serve all strategies $S(R_2, \ell, 2\epsilon)$ at zero cost, and then move the $h - i$ servers back to $R_1$ (cost $\epsilon(h - i)$). Thus, there exists a schedule for serving $c$ iterations with cost $c \cdot 2\epsilon(h - i)$ that ends up in a configuration $(L^{h-i}, h, i)$. By combining this with Lemma 4.6, we get the following corollary.

Corollary 4.7. *The optimal schedule to serve c consecutive iterations of Step 2 using $h + i$ servers in $R$, starting from a configuration $(L^{h-i}, h, i)$ has cost $c \cdot 2\epsilon(h - i)$, and it ends up in a configuration $(L^{h-i}, h, i)$.*

The following definition is going to be helpful for characterizing the changes in the work function during various iterations of Step 2.

*Definition 4.8.* An iteration of Step 2 is called *good* if the following conditions hold:

(i) During the whole iteration, for all $0 \le i < h$ and $0 < j \le h - i$, there is no configuration $C \in (L^{h-i}, R^{h+i})$, which is supported by a configuration $C' \in (L^{h-i-j}, R^{h+i+j})$.
(ii) At the beginning of the iteration, for all $0 \le i < h$, we have that $w(L^{h-i}, R^{h+i}) = w(L^{h-i}, h, i)$ and for all $0 < i' \le h - i$,

$$w(L^{h-i}, h - i', i + i') = w(L^{h-i}, h, i) + i' \cdot \epsilon. \tag{9}$$

Let us get some intuition behind this definition. The first condition implies that during good iterations the work function values of configurations $(L^{h-i}, R^{h+i})$ are not affected by configurations $(L^{h-j}, R^{h+j})$ for $j \neq i$. This makes it easier to argue about $w(L^{h-i}, R^{h+i})$. Moreover, by basic properties, it implies that the WFA does not move servers from $L$ to $R$ (see Observation 4.9 below). Since the WFA has $h$ servers in $R$ when Step 2 starts, our goal is to show that there are too many consecutive good iterations in the beginning of Step 2. This implies that for the largest part of Step 2, the WFA has $h$ servers in $R$. Then, it suffices to get a lower bound on the cost of the WFA during those iterations. The second condition implies that at the beginning of a good phase, any optimal schedule for configurations $(L^{h-i}, R^{h+i})$ is at a configuration $(L^{h-i}, h, i)$.

We now state some basic observations that follow from the definition of a good iteration and will be useful in our analysis.

Observation 4.9. *During a good iteration the WFA does not move any server from $L$ to $R$.*

To see why this is true, consider any time during the iteration when the WFA moves from a configuration $C$ to $C'$, such that $C \in (L^{h-i}, R^{h+i})$, for some $0 \le i < h$. By basic properties of work functions, $C$ is supported by $C'$. Since the iteration is good, $C$ is not supported by any configuration $(L^{h-i-j}, R^{h+i+j})$, i.e., $C' \notin (L^{h-i-j}, R^{h+i+j})$. Thus, the WFA does not move a server from $L$ to $R$.

The following observation comes immediately from Corollary 4.7 and the definition of a good iteration.

Observation 4.10. *Consider a good iteration and let $w, w'$ be the work function at the beginning and at the end of the iteration, respectively. Then, for all $0 \le i \le h$, we have that*

$$w'(L^{h-i}, R^{h+i}) - w(L^{h-i}, R^{h+i}) = 2\epsilon(h-i).$$

Moreover, using Lemma 4.6 for $c = 1$, we get the following observation.

Observation 4.11. *At the end of a good iteration, Equation (9) holds.*

The next lemma gives us a sufficient condition for an iteration to be good.

Lemma 4.12. *Consider an iteration of Step 2 with initial work function $w$. If $w$ satisfies Equation (9) and for all $0 \le i < h$ and $0 < j \le h - i$,*

$$w(L^{h-i}, R^{h+i}) + 3\epsilon \cdot h < w(L^{h-i-j}, R^{h+i+j}) + j,$$

*then the iteration is good.*

Proof. Let $t$ be any time during the iteration with work function $w'$. For any configuration $C \in (L^{h-i}, R^{h+i})$, a possible schedule to serve all requests until time $t$ and end up in $C$ is to simulate the optimal schedule ending up in any configuration $(L^{h-i}, R^{h+i})$, and then moving at most $i \le h$ servers within $R$. The cost of this schedule is at most

$$w'(L^{h-i}, R^{h+i}) + \epsilon h \le w(L^{h-i}, R^{h+i}) + 2\epsilon(h-i) + \epsilon h \le w(L^{h-i}, R^{h+i}) + 3\epsilon h, \tag{10}$$

where the first inequality holds due to Corollary 4.7. Since the right-hand side of Equation (10) is smaller than $w(L^{h-i-j}, R^{h+i+j}) + j \le w'(L^{h-i-j}, R^{h+i+j}) + j$, we get that there is no configuration $C \in (L^{h-i}, R^{h+i})$, which is supported by a configuration $(L^{h-i-j}, R^{h+i+j})$ and thus the iteration is good. □

*Counting Good Iterations.* We now count the number of consecutive good iterations in the beginning of Step 2. For all $0 \le i < h$ and $0 < j \le h - i$, let $D_{i,j} = w(L^{h-i}, R^{h+i}) - w(L^{h-i-j}, R^{h+i+j})$. The next lemma estimates the change in $D_{i,j}$ after a good iteration.

Lemma 4.13. *Consider a good iteration and let $w, w'$ be the work function at the beginning and at the end of the iteration. Then, for all $0 \le i < h$ and $0 < j \le h - i$, we have that*

$$D'_{i,j} = w'(L^{h-i}, R^{h+i}) - w'(L^{h-i-j}, R^{h+i+j}) = w(L^{h-i}, R^{h+i}) - w(L^{h-i-j}, R^{h+i+j}) + 2\epsilon j = D_{i,j} + 2\epsilon j.$$

Proof. By Observation 4.10, we have that

$$\begin{aligned} w'(L^{h-i}, R^{h+i}) - w'(L^{h-i-j}, R^{h+i+j}) &= w(L^{h-i}, R^{h+i}) + 2\epsilon(h-i) - w(L^{h-i-j}, R^{h+i+j}) - 2\epsilon(h-i-j) \\ &= w(L^{h-i}, R^{h+i}) - w(L^{h-i-j}, R^{h+i+j}) + 2\epsilon j. \end{aligned}$$

□

Now, we are ready to estimate the number of consecutive good iterations after Step 2 starts.

Lemma 4.14. *The first $\lceil N - \frac{3h}{2} \rceil$ iterations of Step 2 are good.*

Proof. By Observation 4.11, we have that at the end of a good iteration, Equation (9) holds. Thus, by Lemma 4.12, if at the end of the current iteration $D_{i,j} < j - 3\epsilon h$ for all $0 \le i < h$, $0 < j \le h - i$, then the next iteration is good.

At the beginning of Step 2, $D_{i,j} = -j$, due to Equation (8). From Lemma 4.13, we have that after each good iteration, $D_{i,j}$ increases by $2\epsilon j$. Thus, all iterations will be good, as long as $D_{i,j} < j - 3\epsilon h$. Since

$$\left(\left\lceil N - \frac{3h}{2} \right\rceil - 1\right) \cdot 2\epsilon j < \left(N - \frac{3h}{2}\right) \cdot 2\epsilon j = 2j - 3j\epsilon h \le 2j - 3\epsilon h,$$

we get that at the beginning of the $(\lceil N - \frac{3h}{2} \rceil)$th iteration of Step 2, $D_{i,j} < -j + 2j - 3\epsilon h = j - 3\epsilon h$. Thus, the first $\lceil N - \frac{3h}{2} \rceil$ iterations of Step 2 are good. □

We get the following corollary.

Corollary 4.15. *After $\lceil N - \frac{3h}{2} \rceil$ iterations of Step 2, WFA still has $h$ servers in $R$.*

This holds due to the fact that at the beginning of Step 2, the WFA has $h$ servers in $R$, and from Observation 4.9, we know that the WFA does not move a server from $L$ to $R$ during good iterations.

*Work Function Values for Good Iterations.* Now, we proceed to the characterization of the work function during good iterations. Since we want to estimate the cost of the WFA during iterations when it has $h$ servers at $R$, we focus on configurations $(L^h, R^h)$. Specifically, we need the corresponding versions of Lemmata 4.3, 4.4, and 4.5. Recall that, by definition of a good iteration, the local changes in the values of $w(L^h, R^h)$, define the work function values of all configurations $(L^h, R^h)$.

We begin with the corresponding versions of Lemmata 4.3 and 4.4 for good iterations of Step 2.

Lemma 4.16. *Consider a good iteration of Step 2. Let $t_1$ be the time when $S(R_2, \ell, 2\epsilon)$ ends, for $1 \le \ell \le h$ and $t_2$ the time when $S(R_2, h, 2\epsilon)$ ends. At any time $t \in [t_1, t_2]$, we have that $w(L^h, h - \ell + 1, \ell - 1) = w(L^h, h - \ell, \ell) + \epsilon$.*

Proof. The proof is same as proof of Lemma 4.3 with replacing $(L^{k-i}, R^i)$ by $(L^h, h - i, i)$, scaling distances by $\epsilon$ and noting that by Equation (9), any optimal schedule for configurations $(L^h, R^h)$ starts the iteration from a configuration $(L^h, h, 0)$. □

Lemma 4.17. *Consider a good iteration of Step 2. Let $t_1$ be the time when $S(R_2, \ell, 2\epsilon)$ ends, for $1 \le \ell \le h$ and $t_2$ the time when $S(R_2, h, 2\epsilon)$ ends. At any time $t \in [t_1, t_2]$, we have that, for any $1 \le i < \ell$, $w(L^h, h - i, i) = w(L^h, h - \ell, \ell) + \epsilon(\ell - i)$.*

The proof of this lemma is same as the proof of Lemma 4.4, i.e., by backwards induction on $i$. We just need to replace $(L^{k-i}, R^i)$ by $(L^h, h - i, i)$ and scale all distances by $\epsilon$.

The next two lemmata are the analogues of Lemma 4.5 for good iterations of Step 2, and they characterize the structure of the work function at the end of $S(R_2, \ell, 2\epsilon)$ and $S(R_1, \ell, 2\epsilon)$, for $1 \le \ell \le h$.

Lemma 4.18. *Consider a good iteration during Step 2. At the moment when the execution of $S(R_2, \ell, 2\epsilon)$ ends, for $1 \le \ell \le h$, the following two things hold:*

*(1) For all $0 \le i < \ell$, $w(L^h, h - i, i) = w(L^h, h - \ell, \ell) + \epsilon(\ell - i)$,*
*(2) For all $\ell < i \le h$, $w(L^h, h - i, i) = w(L^h, h - \ell, \ell) + \epsilon(i - \ell)$.*

Proof. For $0 \le i < \ell$, the lemma follows from Lemma 4.17. For $i \ge \ell$, the proof is the same as the proof of Lemma 4.5 by replacing $w(L^{k-i}, R^i)$ by $w(L^h, h - i, i)$ and using the fact that at the beginning of the iteration $w(L^h, h - i, i) = w(L^h, h, 0) + \epsilon \cdot i$. □

Lemma 4.19. *Consider a good iteration during Step 2. At the moment when an execution of $S(R_1, \ell, 2\epsilon)$ ends, for $1 \le \ell \le h$, the following two things hold:*

*(1) For all $0 \le i < \ell$, $w(L^h, i, h - i) = w(L^h, \ell, h - \ell) + \epsilon(\ell - i)$,*
*(2) For all $\ell \le i \le h$, $w(L^h, i, h - i) = w(L^h, \ell, h - \ell) + \epsilon(i - \ell)$.*

Proof. The proof is completely symmetric to the proof of Lemma 4.18. Note that, due to Lemma 4.18, after $S(R_2, h, 2\epsilon)$ ends, i.e., at the beginning of $S(R_1, 1, 2\epsilon)$, we have that

$w(L^h,R^h) = w(L^h,0,h)$ and $w(L^h,h-i,i) = w(L^h,0,h) + \epsilon(h-i)$, or equivalently,

$$w(L^h,i,h-i) = w(L^h,0,h) + \epsilon \cdot i. \tag{11}$$

Note that Equation (11) is symmetric to $w(L^h,h-i,i) = w(L^h,h,0) + \epsilon \cdot i$, which we used in proof of Lemma 4.18, so we can apply the symmetric proof. □

**End of the phase:** We now show that at the end of the phase the work function has the desired structure and that the WFA has all its $2h$ servers in $R$, so a new right-to-left phase may start.

Lemma 4.20. *After $N$ iterations of Step 2, $w(L^{h-i},R^{h+i}) = w(L^0,R^{2h}) + (h-i) = 3h-i$, for all $1 \le i \le h-1$.*

Proof. By basic properties of work functions, $w(L^{h-i},R^{h+i}) \le w(L^0,R^{2h}) + (h-i)$. Assume for contradiction that there exists some $i$ such that $w(L^{h-i},R^{h+i}) < w(L^0,R^{2h}) + (h-i)$. If there are many such $i$'s, then we prove the contradiction for the largest one, and then we can repeat. This assumption implies that for any $j > i$, we have that

$$\begin{aligned} w(L^{h-j},R^{h+j}) + (j-i) &= w(L^0,R^{2h}) + (h-j) + (j-i) \\ &= w(L^0,R^{2h}) + (h-i) > w(L^{h-i},R^{h+i}). \end{aligned}$$

Thus, the configuration $C$ that defines the value of $w(L^{h-i},R^{h+i})$ is not supported by any configuration $(L^{h-j},R^{h+j})$ for $j > i$. Let $\mathcal{S}$ be the schedule that defines the value of $w(L^{h-i},R^{h+i}) = W(C)$. Since $C$ is not supported by any configuration $(L^{h-j},R^{h+j})$ for $j > i$, $\mathcal{S}$ uses exactly $h+i$ servers in $R$. We now evaluate the cost of $\mathcal{S}$. Moving $h+i$ servers from $L$ to $R$ costs $h+i$. Serving Step 1 using $h+i$ servers in $R$ costs 0. By Corollary 4.7, the optimal way to serve $N$ iterations of Step 2 using $h+i$ servers in $R$ has cost $2N \cdot \epsilon(h-i) = 2(h-i)$. Overall, we get that $w(L^{h-i},R^{h+i}) = h + i + 2(h-i) = 3h-i$.

However, we have that $w(L^0,R^{2h}) = 2h$ for the whole phase. Thus,

$$w(L^{h-i},R^{h+i}) = 3h - i = w(L^0,R^{2h}) + (h-i),$$

contradiction. □

By setting $i' = h-i$, Lemma 4.20 gives us that

$$w(L^{i'},R^{2h-i'}) = w(L^0,R^{2h}) + i', \text{ for } 1 \le i' \le h. \tag{12}$$

Moreover, by setting $i' = 2h-i$ in Equation (7), we get that for $h < i' \le 2h$,

$$w(L^{i'},R^{2h-i'}) = w(L^h,R^h) + i' - h = w(L^0,R^{2h}) + h + i' - h = w(L^0,R^{2h}) + i', \tag{13}$$

where the first equality holds due to Equation (12). From Equations (12) and (13), we get that after $N$ iterations of Step 2,

$$w(L^{i'},R^{2h-i'}) = w(L^0,R^{2h}) + i', \text{ for all } 1 \le i' \le 2h. \tag{14}$$

Note that $w(L^0,R^{2h})$ does not change during the whole phase, since exactly $2h$ points of $R$ are requested. Thus, Equation (14) holds at the end of the phase and then a new right-to-left phase may start, satisfying the assumption about the initial work function. Last, to see that at the end of the phase the WFA has all its $2h$ servers in $R$, assume that after $N$ iterations of Step 2, it is in some configuration $(L^{i'},R^{2h-i'})$ for some $0 \le i' \le (h-1)$. Since Equation (14) holds at the end of the phase, during the next iteration, the WFA moves to the configuration $(L^0,R^{2h})$.

*Bounding Costs.* We now bound the online and offline costs. We begin with Step 1.

Lemma 4.21. *During the time when the WFA uses $\ell < h$ servers in $R$, it incurs a cost of at least $(2-2\epsilon') \cdot \ell$.*

Proof. By construction of the phase, when the execution of $S(R_1, \ell+1, 2)$ starts, the WFA has $\ell$ servers in $R$. We evaluate the cost of the WFA during the part of $S(R_1, \ell+1, 2)$ when it has $\ell$ servers in $R$. Let $t$ be the time when the WFA moves its $(\ell+1)$th server to $R$. Let $w$ and $w'$ be the work functions at the beginning of $S(R_1, \ell+1, 2)$ and at time $t$, respectively. We have that

$$w'(L^{k-\ell-1}, R^{\ell+1}) = w(L^{k-\ell-1}, R^{\ell+1}) = w(L^{k-\ell}, R^{\ell}) + 1, \tag{15}$$

where the first equality comes from the fact that there are exactly $\ell+1$ points of $R$ requested since the beginning of the phase and the second by Lemma 4.5. Let $C$ be the configuration of the WFA at time $t'-1$. At time $t$, the WFA moves a server by a distance of 1, thus by basic properties of work functions, we have that $w'(C) = w'(L^{k-\ell-1}, R^{\ell+1}) + 1$. This combined with Equation (15) gives that

$$w'(C) = w(L^{k-\ell}, R^{\ell}) + 2. \tag{16}$$

Let $X \in (L^{k-\ell}, R^{\ell})$ be a configuration such that (i) $w'(X) = w'(L^{k-\ell}, R^{\ell})$ and (ii) $X, C$ have their $k-\ell$ servers of $L$ at the same points[3]. Note that $X, C \in (L^{k-\ell}, \ell, 0)$, since $R_1$ is the only subtree of $R$ where requests have been issued. Since both $X$ and $C$ have $\ell$ servers in $R_1$, and only $\ell+1$ leaves of $R_1$ have been requested, we get that $X$ and $C$ can differ in at most one point. In other words, $d(X, C) \leq \epsilon'$. By basic properties of work functions, we get that

$$w'(C) \leq w'(X) + d(X, C) \leq w'(L^{k-\ell}, R^{\ell}) + \epsilon'. \tag{17}$$

From Equations (16) and (17), we get that

$$w'(L^{k-\ell}, R^{\ell}) - w(L^{k-\ell}, R^{\ell}) \geq w'(C) - \epsilon' - (w'(C) - 2) = 2 - \epsilon'. \tag{18}$$

Let $\mathcal{S}_\ell$ be the optimal schedule to serve all requests from the beginning of $S(R_2, \ell+1, 2\epsilon)$ until time $t$, using $\ell$ servers starting at $t_1, \ldots, t_\ell$. From Equation (18), we have that $\text{cost}(\mathcal{S}_\ell) \geq 2 - \epsilon'$. All requests are located in $\ell+1$ leaves of an elementary subtree (which is equivalent to the paging problem) in an adversarial way. It is well known (see, e.g., Borodin and El-Yaniv (1998)) that the competitive ratio of any online algorithm for the paging problem for such a sequence is at least $\ell$, i.e., it has cost at least $cost(\mathcal{S}_\ell) - \ell \cdot \epsilon'$, where $\ell \cdot \epsilon'$ is the allowed additive term. This implies that the cost incurred by the WFA is at least

$$\ell \cdot \text{cost}(\mathcal{S}_\ell) - \ell \cdot \epsilon' \geq \ell(2 - \epsilon') - \ell \cdot \epsilon' = (2 - 2\epsilon') \cdot \ell. \qquad \square$$

Lemma 4.1. *(restated) During Step 1,* ALG $\geq h^2 - \epsilon' \cdot h(h-1)$ *and ADV*$= h$.

Proof. Clearly, ADV$= h$; the adversary moves $h$ servers by distance 1 and then serves all requests at zero cost. By Lemma 4.21, we get that WFA incurs a cost of at least $\sum_{\ell=1}^{h-1}(2-2\epsilon') \cdot \ell$ inside $R$. Moreover, it pays a movement cost of $h$ to move its $h$ servers from $L$ to $R$. Overall, we get that the online cost during Step 1 is $(2-2\epsilon')\sum_{\ell=1}^{h-1}\ell + h = 2\sum_{\ell=1}^{h-1}\ell + h - 2\epsilon'\sum_{\ell=1}^{h-1}\ell = h^2 - \epsilon' \cdot h(h-1)$. $\square$

We now focus on Step 2. We charge the WFA only for the first $\lceil N - \frac{3h}{2} \rceil$ iterations (when it uses $h$ servers in $R$, due to Corollary 4.15), plus the movement cost of the extra servers from $L$ to $R$.

Lemma 4.22. *Consider a good iteration (see Definition 4.8) of Step 2 such that the WFA uses only $h$ servers in $R$. During the time interval that WFA serves requests in subtree $R_1$ or $R_2$ using $\ell$ servers, it incurs a cost of at least $2(\epsilon - \epsilon')\ell$.*

Proof. The proof is similar to the proof of Lemma 4.21, with the following modifications. We scale distances by $\epsilon$, since now the servers move from $R_1$ to $R_2$ (distance $\epsilon$) instead of moving

[3]Such a configuration always exists, since there are no requests at $L$, hence the locations of the servers in $L$ does not affect the work function values.

from $L$ to $R_1$ (distance 1). We charge the WFA for serving requests in $R_2$ using $\ell$ servers during executions of $S(R_2, \ell+1, 2\epsilon)$, using Lemma 4.18 (instead of Lemma 4.5) and replacing $(L^{k-i}, R^i)$ by $(L^h, h-i, i)$. Similarly, we charge the WFA for serving requests in $R_1$ using $\ell$ servers during executions of $S(R_1, \ell+1, 2\epsilon)$ using Lemma 4.19. □

Lemma 4.23. *Consider a good iteration of Step 2 such that the WFA uses only $h$ servers in $R$ to serve the requests. The cost of the WFA in such an iteration is at least $2\epsilon \cdot h^2 - 2\epsilon' \cdot (h-1) \cdot h$.*

Proof. From Lemma 4.22, we get that the cumulative cost for all executions of $S$ in $R_2$ is $\sum_{\ell=1}^{h-1} 2(\epsilon - \epsilon')\ell + \epsilon \cdot h$, where the last $\epsilon \cdot h$ term counts the cost of moving $h$ servers from $R_1$ to $R_2$. Similarly, the WFA incurs the same cumulative cost during executions of $S$ in $R_1$. We get that the total cost of the WFA during the iteration is at least

$$2 \cdot \left( \sum_{\ell=1}^{h-1} 2\left(\epsilon - \epsilon'\right)\ell + \epsilon \cdot h \right) = 2\epsilon \cdot h + 4 \sum_{\ell=1}^{h-1} \epsilon \cdot \ell - 4 \sum_{\ell=1}^{h-1} \epsilon' \cdot \ell,$$

$$2\epsilon\left(h + (h-1)\,h\right) - 2\epsilon' \cdot (h-1) \cdot h = 2\epsilon \cdot h^2 - 2\epsilon' \cdot (h-1) \cdot h. \qquad \square$$

Lemma 4.2 (Restated). *During Step 2,* ALG $\geq 2h^2 + h - 3\epsilon h^3 - 2\frac{\epsilon'}{\epsilon} \cdot (h-1) \cdot h$ *and* ADV $= (2+\epsilon)h$.

Proof. The second step consists of $N + 1 = 1/\epsilon + 1$ iterations, where in each one of them the adversary incurs a cost of $2\epsilon \cdot h$. Thus, the offline cost is $(2+2\epsilon)h$.

Let us now count the online cost during Step 2. By Corollary 4.15, there are at least $\lceil N - \frac{3h}{2} \rceil$ iterations such that the WFA has $h$ servers in $R$. By Lemma 4.23, we get that during each one of those iterations, the online cost is at least $2\epsilon \cdot h^2 - 2\epsilon' \cdot (h-1) \cdot h$. For the rest of Step 2, the WFA incurs a cost of at least $h$, as it moves $h$ servers from $L$ to $R$. We get that overall, during Step 2, the cost of WFA is at least

$$\left(N - \frac{3h}{2}\right) \cdot (2\epsilon \cdot h^2 - 2\epsilon' \cdot (h-1) \cdot h) + h = 2N\epsilon \cdot h^2 - 2N\epsilon' \cdot (h-1) \cdot h - 3\epsilon h^3 + 3\epsilon' h^2 (h-1) + h$$

$$\geq 2h^2 + h - 3\epsilon h^3 - 2\frac{\epsilon'}{\epsilon} \cdot (h-1) \cdot h. \qquad \square$$

*Remaining Proofs.* We now prove Lemma 4.3, whose proof was omitted earlier. First, we state an observation that follows directly from the definition of the left-to-right phase.

Observation 4.24. *Let $t$ be any time during a left-to-right phase with work function $w$. If $w(L^{k-i}, R^i) = w(L^{k-j}, R^j) + (j-i)$ for some $0 \leq i < j \leq k$, then for all $j'$ such that $i \leq j' \leq j$, we have that $w(L^{k-j'}, R^{j'}) = w(L^{k-j}, R^j) + (j-j')$.*

To see why this is true, note that by basic properties $w(L^{k-j'}, R^{j'}) \leq w(L^{k-j}, R^j) + (j-j')$. Moreover, if strict inequality holds, we get that $w(L^{k-j'}, R^{j'}) + (j'-i) < w(L^{k-j}, R^j) + (j-j') + (j'-i) = w(L^{k-j}, R^j) + (j-i) = w(L^{k-i}, R^i)$. We get that $w(L^{k-i}, R^i) - w(L^{k-j'}, R^{j'}) > j'-i$, a contradiction.

Lemma 4.3 (Restated). *Consider a left-to-right phase. At any time after the end of strategy $S(R_1, \ell, 2)$, for $1 \leq \ell \leq h$, we have that $w(L^{k-\ell+1}, R^{\ell-1}) = w(L^{k-\ell}, R^\ell) + 1$.*

Proof. We prove the lemma by induction. For the base case $\ell = 1$, the lemma is clearly true, since any schedule serving requests uses at least one server at $R$. We prove the lemma for $2 \leq \ell \leq h$, assuming that it is true for all values $1, \ldots, \ell - 1$.

Let $t$ be any time after the end of strategy $S(R_1, \ell, 2)$, for $1 < \ell \leq h$ with work function $w$. By basic properties of work functions, $w(L^{k-\ell+1}, R^{\ell-1}) \leq w(L^{k-\ell}, R^{\ell}) + 1$. We will assume that $w(L^{k-\ell+1}, R^{\ell-1}) < w(L^{k-\ell}, R^{\ell}) + 1$ and obtain a contradiction. The lemma then follows.

Assume that $w(L^{k-\ell+1}, R^{\ell-1}) < w(L^{k-\ell}, R^{\ell}) + 1$. Let $\mathcal{S}_{\ell-1}$ and $\mathcal{S}_{\ell}$ be the schedules that define the values of $w(L^{k-\ell+1}, R^{\ell-1})$ and $w(L^{k-\ell}, R^{\ell})$, respectively. Let $C$ be a configuration $(L^{k-\ell+1}, R^{\ell-1})$ such that $w(C) = w(L^{k-\ell+1}, R^{\ell-1})$. By Observation 4.24, $C$ is not supported by any configuration $(L^{k-i}, R^{i})$, for $i \geq \ell$. Also, by the inductive hypothesis, $C$ is not supported by configurations $(L^{k-i}, R^{i})$ for $i < \ell - 1$. We get that $C$ is in the support of $w$. Without loss of generality $\mathcal{S}_{\ell-1}$ is lazy, i.e., it moves a server only to serve a request (this is a standard property, explained in the proof of Theorem 1.4). Thus, from the beginning of the phase until time $t$, $\mathcal{S}_{\ell-1}$ never moves a server from $R$ to $L$. We get that $\mathcal{S}_{\ell-1}$ uses exactly $\ell - 1$ servers in $R$. However, by construction of the phase, $\mathcal{S}_{\ell}$ has at least $\ell$ servers in $R$ during the execution of $S(R_1, \ell, 2)$.

We now compare the costs of $\mathcal{S}_{\ell-1}$ and $\mathcal{S}_{\ell}$. Before $S(R_1, \ell, 2)$ starts, the schedules are the same, since there are exactly $\ell - 1$ points of $R$ requested. During $S(R_1, \ell, 2)$, $\mathcal{S}_{\ell}$ incurs a cost of 1 to move a server from $L$ to $R$ and then serves all requests at zero cost, while $\mathcal{S}_{\ell-1}$ incurs a cost of at least 2 (by construction of the strategy). When $S(R_1, \ell, 2)$ ends, the set of leaves of $R_1$ occupied by servers of $\mathcal{S}_{\ell-1}$ is a subset of the leaves occupied by servers of $\mathcal{S}_{\ell}$. Thus, the cost incurred by $\mathcal{S}_{\ell}$ after the end of $S(R_1, \ell, 2)$ is smaller or equal to the cost incurred by $\mathcal{S}_{\ell-1}$ during the same time interval. Overall, we get that $\text{cost}(\mathcal{S}_{\ell-1}) \geq \text{cost}(\mathcal{S}_{\ell}) + 1$, i.e., $w(L^{k-\ell+1}, R^{\ell-1}) \geq w(L^{k-\ell}, R^{\ell}) + 1$, a contradiction. □

## 5 CONCLUDING REMARKS

Several intriguing open questions remain, and we list some of them here.

- Is the dependence on $d$ in Theorem 1.3 necessary? While Theorem 1.4 gives a separation between depth-1 and depth-2 HSTs, it is unclear to us whether a lower bound that increases substantially with depth is possible. Note that a lower bound of $g(d)$ for depth $d$, where $g(d) \to \infty$ as $d \to \infty$ (provided that $h$ is large enough), would be very surprising. This would imply that there is no $O(1)$-competitive algorithm for general metric spaces.
- Can we get an $o(h)$-competitive algorithm for other metric spaces? An interesting metric is the line: Both DC and WFA have competitive ratio $\Omega(h)$ in the line, and we do not even know any good candidate algorithm. Designing an algorithm with such a guarantee would be very interesting.

## A ALGORITHM

### A.1 Proofs of Lemmas from Section 3

Proof of Lemma 3.7 The proof is quite similar to the proof of Lemma 3.6. Instead of using the fact that $\delta \geq \beta^d = \beta \cdot \beta^{d-1} = \beta \cdot \beta_d$, we now crucially use the fact that $\beta_\ell = \beta \cdot \beta_{\ell-1}$. However, now we have to be more careful with the counting of the servers of the adversary.

For each $s \in D$, let $\ell(s)$ be the level of the edge $e$ containing $s$. Similarly to the the proof of Lemma 3.6, we have $k_s \leq \lfloor \beta_{\ell(s)} \cdot h_s \rfloor \leq \beta_{\ell-1} \cdot h_s$ for each server $s \in D$, since $\ell(s) \leq \ell - 1$. Recall that we assume that the adversary has already served the request. Let $a$ be the server of the adversary located at the requested point. Since no online servers reside in the path between $q$ and the requested point, $a$ does not belong to $T_s$ for any $s \in D \cup E$. Therefore, we have $\sum_{s \in D} h_s \leq \sum_{s \in (D \cup E)} h_s \leq h_q^- - 1$. We get that

$$\sum_{s \in D} k_s \leq \sum_{s \in D} \beta_{\ell-1} \cdot h_s \leq \beta_{\ell-1}(h_q^- - 1). \tag{19}$$

To finish the proof of the first inequality, observe that our assumption that $k_q^- \geq \lfloor \beta_\ell h_q^- \rfloor$ implies

$$k_q^- \geq \lfloor \beta_\ell h_q^- \rfloor \Rightarrow \beta_\ell h_q^- \leq k_q^- + 1 \Leftrightarrow h_q^- \leq \frac{k_q^- + 1}{\beta_\ell}.$$

Therefore, from Equation (19), we have

$$\sum_{s \in D} k_s \leq \beta_{\ell-1}\left(h_q^- - 1\right) \leq \beta_{\ell-1}\left(\frac{k_q^- + 1}{\beta_\ell} - 1\right) = \frac{\beta_{\ell-1}}{\beta_\ell} k_q^- + \frac{1}{\beta} - \beta_{\ell-1} \leq \frac{1}{\beta} k_q^-.$$

For the second inequality, note that $\frac{1}{\gamma} = (1 - \frac{1}{\beta})$. Since $k_q^- = \sum_{s \in D} k_s + \sum_{s \in E} k_s$, we have

$$\sum_{s \in E} k_s \geq k_q^- - \frac{1}{\beta} k_q^- = \frac{1}{\gamma} k_q^-. \qquad \square$$

### A.2 Algorithm for Bounded-Diameter Trees

Since Algorithm 1 works for depth-$d$ trees with arbitrary edge lengths, we can embed any diameter-$d$ tree into a depth-$d$ tree with a very small distortion by adding fake paths of short edges to all nodes.

More precisely, let $T$ be a tree of diameter $d$ with arbitrary edge lengths, and let $\alpha$ be the length of the shortest edge of $T$ (for any finite $T$ such $\alpha$ exists). We fix $\epsilon > 0$ a small constant. We create an embedding $T'$ of $T$ as follows. We choose the root $r$ arbitrarily, and to each node $v \in T$ such that the path from $r$ to $v$ contains $\ell$ edges, we attach a path containing $d - \ell$ edges of total length $\epsilon\alpha/2$. The leaf at the end of this path we denote $v'$. We run Algorithm 1 in $T'$ and each request at $v \in T$, we translate to $v' \in T'$. We maintain the correspondence between the servers in $T$ and the servers in $T'$, and the same server, which is finally moved to $v'$ by Algorithm 1, we also use to serve the request $v \in T$.

For the optimal solutions on $T$ and $T'$, we have $(1+\epsilon)OPT(T) \geq OPT(T')$, since any feasible solution in $T$ can be converted to a solution in $T'$ with cost at most $(1+\epsilon)$ times higher. By Theorem 3.8, we know that the cost of Algorithm 1 in $T'$ is at most $R \cdot OPT(T')$, for $R = \Theta(d \cdot \gamma^{d+1})$, and therefore we have $ALG(T') \leq (1+\epsilon)R \cdot OPT(T)$.

## B LOWER BOUND FOR WFA ON THE LINE

The lower bound of Section 4.2 for the WFA can also be applied on the line. The lower bound strategy is the same as the one described for depth-3 HSTs, we just need to replace subtrees by line segments.

More precisely, the lower bound strategy is applied in an interval $I$ of length 1. Let $L$ and $R$ be the leftmost and rightmost regions of I, of length $\epsilon/2$, for $\epsilon \ll 1$. Similarly, $L_1, L_2$ and $R_1, R_2$ are smaller regions inside $L$ and $R$, respectively. Again, the distance between $L_1$ and $L_2$ ($R_1$ and $R_2$, respectively) is much larger than their length. In each of those regions, we focus on $h$ points, where the distance between two consecutive points is $\epsilon' \ll \epsilon$. Whenever the adversary moves to such a region, it places its servers those $h$ equally spaced points inside it. Similarly to the proof of Theorem 1.2, we can get a lower bound $h + 1/3$ on the competitive ratio of the WFA on the line when $k = 2h$ (the changes affect only the dependence on terms involving $\epsilon$ and $\epsilon'$, which are negligible).

Interestingly, the lower bound obtained by this construction for the line has a matching upper bound. As it was observed in Bansal et al. (2018), results from Bartal and Koutsoupias (2004) and Koutsoupias (1999) imply that for the line the cost of WFA with $k$ servers $\text{WFA}_k$ is at most $(h+1)\text{OPT}_h - \text{OPT}_k + \text{const}$ , where $\text{OPT}_i$ is the optimal cost using $i$ servers. Briefly, this upper

bound holds for the following reason: In Koutsoupias (1999) it was shown that in general metrics, $\mathrm{WFA}_k \leq 2h\mathrm{OPT}_h - \mathrm{OPT}_k + \text{const}$. However, if we restrict our attention to the line, using the result of Bartal and Koutsoupias (2004), we can get that

$$\mathrm{WFA}_k \leq (h+1)\mathrm{OPT}_h - \mathrm{OPT}_k + \text{const}. \tag{20}$$

See Bansal et al. (2018) for more details.

In Theorem 1.2, we showed a lower bound $(h + 1/3)\mathrm{OPT}_h$ on $\mathrm{WFA}_k$. We now show that this construction matches the upper bound of Equation (20). In particular, it suffices to show that $(h + 1/3)\mathrm{OPT}_h$ goes arbitrary close to $(h+1)\mathrm{OPT}_h - \mathrm{OPT}_k$, or equivalently,

$$\frac{2\mathrm{OPT}_h}{3} \to \mathrm{OPT}_k. \tag{21}$$

As we showed in the proof of Theorem 1.2, for every phase of the lower bound strategy, $\mathrm{OPT}_h = (3 + 2\epsilon) \cdot h$. Moreover it is clear that $\mathrm{OPT}_k = 2h$; during a phase the minimum work function value using $k = 2h$ servers increases by exactly $2h$. We get that

$$\frac{2\mathrm{OPT}_h}{3} = \frac{2 \cdot (3+2\epsilon) \cdot h}{3} = 2h\frac{3+2\epsilon}{3} \to 2h = \mathrm{OPT}_k.$$

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments, which helped us improve the presentation and simplify the analysis of our algorithm.

## REFERENCES

Nikhil Bansal, Marek Eliás, Lukasz Jez, and Grigorios Koumoutsos. 2017. The $(h, k)$-server problem on bounded depth trees. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*. 1022–1037.

Nikhil Bansal, Marek Eliás, Lukasz Jez, Grigorios Koumoutsos, and Kirk Pruhs. 2018. Tight bounds for double coverage against weak adversaries. *Theory Comput. Syst.* 62, 2 (2018), 349–365.

Yair Bartal. 1996. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS'96)*. 184–193.

Yair Bartal and Elias Koutsoupias. 2004. On the competitive ratio of the work function algorithm for the k-server problem. *Theor. Comput. Sci.* 324, 2–3 (2004), 337–345.

Allan Borodin and Ran El-Yaniv. 1998. *Online Computation and Competitive Analysis*. Cambridge University Press.

Marek Chrobak, Howard J. Karloff, Thomas H. Payne, and Sundar Vishwanathan. 1991. New results on server problems. *SIAM J. Discrete Math.* 4, 2 (1991), 172–181.

Marek Chrobak and Lawrence L. Larmore. 1991. An optimal on-line algorithm for k-servers on trees. *SIAM J. Comput.* 20, 1 (1991), 144–148.

Christian Coester, Elias Koutsoupias, and Philip Lazos. 2017. The infinite server problem. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP'17)*. 14:1–14:14.

Bala Kalyanasundaram and Kirk Pruhs. 2000. Speed is as powerful as clairvoyance. *J. ACM* 47, 4 (July 2000), 617–643.

Elias Koutsoupias. 1999. Weak adversaries for the k-server problem. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS'99)*. 444–449.

Elias Koutsoupias and Christos H. Papadimitriou. 1995. On the k-server conjecture. *J. ACM* 42, 5 (1995), 971–983.

Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. 1990. Competitive algorithms for server problems. *J. ACM* 11, 2 (1990), 208–230.

Daniel Dominic Sleator and Robert Endre Tarjan. 1985. Amortized efficiency of list update and paging rules. *Commun. ACM* 28, 2 (1985), 202–208.

Neal E. Young. 1994. The k-server dual and loose competitiveness for paging. *Algorithmica* 11, 6 (1994), 525–541.

Received September 2017; revised September 2018; accepted November 2018