The Online Metric Matching Problem for Doubling Metrics

Anupam Gupta[1,⋆] and Kevin Lewi[2,⋆⋆]

[1] Computer Science Department, Carnegie Mellon University
[2] Computer Science Department, Stanford University

Abstract. In the online minimum-cost metric matching problem, we are given an instance of a metric space with k servers, and must match arriving requests to as-yet-unmatched servers to minimize the total distance from the requests to their assigned servers. We study this problem for the line metric and for doubling metrics in general. We give $O(\log k)$-competitive randomized algorithms, which reduces the gap between the current $O(\log^2 k)$-competitive randomized algorithms and the constant-competitive lower bounds known for these settings.

We first analyze the "harmonic" algorithm for the line, that for each request chooses one of its two closest servers with probability inversely proportional to the distance to that server; this is $O(\log k)$-competitive, with suitable guess-and-double steps to ensure that the metric has aspect ratio polynomial in k. The second algorithm embeds the metric into a random HST, and picks a server randomly from among the closest available servers in the HST, with the selection based upon how the servers are distributed within the tree. This algorithm is $O(1)$-competitive for HSTs obtained from embedding doubling metrics, and hence gives a randomized $O(\log k)$-competitive algorithm for doubling metrics.

1 Introduction

In the online minimum-cost metric matching problem, the input is a metric space (V, d) with k pre-specified *servers* $S \subseteq V$. The requests $R = r_1, r_2, \ldots, r_k$ (with each $r_i \in V$) arrive online one-by-one; upon arrival each request must be immediately and irrevocably matched to an as-yet-unmatched server. The cost of matching request r to server $f(r) \in S$ is the distance $d(r, f(r))$ in the underlying metric space. The goal is to find a matching f that approximately minimizes the total cost $\sum_i d(r_i, f(r_i))$. We study the problem in the framework of competitive analysis, comparing the cost of our algorithm's matching to the cost of the best offline matching from R to S. (This minimum cost bipartite perfect matching problem can be easily solved offline.)

The online problem was introduced in the early 1990's by Kalyanasundaram and Pruhs [4], and by Khuller, Mitchell, and Vazirani [6]. Both papers gave

⋆ Supported in part by NSF awards CCF-0964474 and CCF-1016799.

⋆⋆ Work done when at the Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15217.

A. Czumaj et al. (Eds.): ICALP 2012, Part I, LNCS 7391, pp. 424–435, 2012.

deterministic $2k-1$-competitive algorithms, which is the best possible even when the metric is a star with k leaves with the servers at the leaves. For the star example, any *randomized* algorithm must be $\Omega(H_k)$-competitive, and the natural randomized greedy algorithm is indeed $O(H_k)$-competitive, where H_k is the k^{th} harmonic number. In 2006, Meyerson et al. [9] showed that the randomized greedy algorithm, which assigns to a uniformly random closest server, is $O(\log k)$-competitive when the metric is a α-hierarchically well-separated tree (α-HST) with suitably large separation α between levels, namely with $\alpha = \Omega(\log k)$. This implies an $O(\log^3 k)$-competitive randomized algorithm for general metrics using randomized embeddings into HSTs [2]. Bansal et al. [1] gave a different algorithm which is $O(\log k)$-competitive on 2-HSTs, resulting in an $O(\log^2 k)$ competitive algorithm on general metrics. It remains an open problem to close the gap between $O(\log^2 k)$ and $\Omega(\log k)$ for general metrics.

The gap is even worse when we consider natural special classes of metrics such as the line, or grids, or doubling metrics. For points on the line, the best *deterministic* lower bound is only 9.001 [3] (with the randomized lower bound being even weaker), and no algorithms better than those that apply to general metrics are known for neither the line nor doubling metrics, both in the deterministic and randomized settings.

Results and Techniques. In this paper, we give randomized algorithms for restricted classes of metrics. In particular, we show $O(\log k)$-competitive randomized algorithms for the online metric matching problem on the line metric and on doubling metrics. (For the rest of this section, we assume that the *aspect ratio* of the metric, namely the maximum-to-minimum distance ratio, is $O(k^3)$—this can be achieved with only a constant factor loss using the guess-and-double framework, details in the full version.)

Our first algorithm is the natural randomized *Harmonic* algorithm: letting s_L and s_R be the closest available left and right servers to the current request r, we assign r to s_L with probability

$$\frac{1/d(r,s_L)}{1/d(r,s_L)+1/d(r,s_R)} = \frac{d(r,s_R)}{d(s_L,s_R)},$$

and to s_R with the remaining probability. If $d_{\max}$ and $d_{\min}$ are the largest and smallest distances between any two servers, we show:

Theorem 1. *The* Harmonic *algorithm is $O(\log \frac{d_{\max}}{d_{\min}})$-competitive for the line. Hence, with using guess-and-double to ensure $d_{\max}/d_{\min} = O(k^3)$, we get an $O(\log k)$-competitive algorithm for the line.*

Our proof uses a coupling argument: we consider two runs of the *Harmonic* algorithm, the first starting with some set S of servers and the second with the set $(S \cup \{s_1\} \setminus \{s_2\})$—i.e., differing from S in exactly one server. We show that the expected difference in cost between these two runs of *Harmonic* is $O(\log \frac{d_{\max}}{d_{\min}}) \cdot d(s_1, s_2)$. Now, if we construct a sequence of hybrid algorithms (each of which first follows the optimal algorithm, and at some point switches to the *Harmonic* algorithm), we can use the coupling argument to compare the

runs of adjacent pairs on these sequences to bound the difference between our algorithm and Opt. This idea is similar to the path-coupling idea of Bubley and Dyer, used to show mixing of Markov chains.

Our second algorithm *Random-Subtree* generalizes to the broader class of doubling metrics. It first embeds the metric into a random distance-preserving Δ-degree α-HST, and then runs a certain randomized greedy algorithm on this new instance, where Δ and α are constants that depend on the doubling dimension. At first glance, using a $O(1)$-HST seems bad, since Meyerson et al. showed that their randomized greedy algorithm requires a large separation α in the HST. However, we avoid the lower bound (a) by using the fact that the bad examples require large degrees, whereas HSTs obtained from the line and doubling metrics have small degree, and (b) by altering the randomized greedy algorithm slightly (in a way we will soon describe). At a high level, we show that if a metric can be embedded into an α-HST where each vertex has at most Δ children, our randomized algorithm is $O(H_\Delta/\epsilon)$-competitive on such an HST, so long as $\alpha \geq (1+\epsilon)H_\Delta$. (See Theorem 5 for a precise statement.) Since all doubling metrics admit such embeddings (for values of Δ, α depending only on the doubling dimension) with $O(\log k)$ expected stretch, we get:

Theorem 2. *The randomized algorithm* Random-Subtree *is* $O(\log k)$*-competitive for online metric matching on doubling metrics, and hence also for the line.*

The improvement from $O(\log^3 k)$ in [9] (for general metrics) to $O(\log k)$ (for doubling metrics) is due to both the nature of doubling metrics and the HSTs arising from them, and also due to our algorithm *Random-Subtree* differing from that of [9]. Instead of picking a *uniformly* random available server closest to the request in the HST, we use the following procedure: starting off at the lowest ancestor of the request that contains an available server, our algorithm repeatedly moves us to a uniformly random subtree of this node that has an available server until we reach a leaf/server. Note that our process does not pick a random closest server, but biases towards available servers in subtrees with few available servers. This results in such subtrees being empty earlier, which in turn results in fewer choices higher up in the tree for future requests. Our potential function-based analysis refines the one from [9] by using this property.

The rest of the paper is as follows. We present some notation and preliminaries in Section 2. The *Harmonic* algorithm is analyzed in Section 3, and the *Random-Subtree* algorithm presented and analyzed in Section 4. Due to lack of space, many proofs are deferred to the full version. Also in the full version, we present a third algorithm that is $O(\log k)$ competitive for matching on the line. This algorithm also embeds the line into a random HST, but then runs deterministically on the resulting HST to give this guarantee.

Other Related Work. The paper [5, Section 2.2] gave a lower bound of 9 for deterministic algorithms on the line via a reduction from the so-called cow-path problem; they conjectured this lower bound was tight for the line, which was disproved in [3]. [5] also conjectured that the work function algorithm (see,

e.g., [8]) obtains an $O(1)$-competitive ratio on the line; this was disproved by Koutsoupias and Nanavati [7], who showed an $\Omega(\log k)$ lower bound (and an $O(k)$ upper bound) for the work function algorithm. There is no algorithm, either randomized or deterministic, currently conjectured to be $O(1)$-competitive.

2 Notation and Preliminaries

An instance of the problem is given by a metric (V, d) with servers at $S \subseteq V$, where $|S| = k$. As mentioned in [9], we can assume without loss of generality that all requests also arrive at vertices in S (with only a constant factor loss in the competitive ratio). Hence, in the rest of the paper, we assume that $S = V$, and hence $|V| = |S| = k$. Moreover, we assume there is only one server at each node, as this is only for ease of exposition and the algorithms easily extend to multiple servers at nodes.

An α-HST (Hierarchically well-Separated Tree) is defined as a rooted tree where all edges at depth i have weight c/α^i for some fixed constant c. Here, the edges at depth 0 are those incident to the root, etc. An HST is Δ-ary if each node has at most Δ children. For the case of the line metric, we assume that the aspect ratio of the points containing the servers (which, recall, is defined as $\frac{\max_{x,y \in S} d(x,y)}{\min_{x,y \in S} d(x,y)}$) is $O(k^3)$; in the full version, we show that this loses only a constant factor in the competitiveness. This allows us to embed these points into distributions of dominating *binary* 2-HSTs with expected stretch $O(\log k)$. Furthermore, for HSTs that are constructed from the line, we refer to the *width* of a tree as the maximum line-distance between any two points within the tree. For doubling metrics we cannot make such a general assumption on the aspect ratio; however, by suitably guessing the value of Opt and running the HST construction algorithms only for the top $O(\log k)$ levels, one can still give a reduction to the problem on bounded-degree HSTs with only an $O(\log k)$-expected loss. (Details in the full version.)

For a node a of a tree, let $T(a)$ represent the subtree rooted at a. Also, define the *level of* a to be the maximum number of edges on a path from a to a leaf of $T(a)$. When referring to servers to be assigned by requests, we will refer to servers that have not yet been assigned to as "available", "free", or "unassigned". We will use Opt to denote both the optimum matching as well as its cost.

3 The Harmonic Algorithm for the Line

To prove the performance guarantee for the *Harmonic* algorithm given by Theorem 1, we first give a lemma which analyzes the expected difference in cost between running *Harmonic* on all the requests and running the optimal algorithm *for just the first step*, and *Harmonic* thenceforth. Then, we show that using this bound in a "hybrid argument" proves Theorem 1. (This is essentially the path-coupling idea of Bubley and Dyer.) For a request sequence $\sigma = r_1, \ldots, r_k$, let g_σ be the matching obtained by assigning $r_1, \ldots, r_k$ using *Harmonic*. Let

$N(r_t)$ be the set of available neighboring servers to r_t—those which are closest to r_t on the left or right and available at time t^-. Define h_σ to be a matching obtained by first matching r_1 to an given server $s_1 \in N(r_1)$, and then using *Harmonic* to assign $r_2, \ldots, r_k$. We will use G for the algorithm producing g_σ and H for h_σ.

Lemma 3 (Hybrid Lemma). *If distances between servers are in* $\mathbb{Z} \cap [0, \Gamma]$,

$$E_G\left[\sum_{i=1}^{k} d(r_i, g_\sigma(r_i))\right] - E_H\left[\sum_{i=1}^{k} d(r_i, h_\sigma(r_i)\right] \leq O(\log \Gamma) \cdot d(r_1, s_1).$$

In other words, the expected cost of G for any request sequence is at most the expected cost of H on the same request sequence *plus* $O(\log \Gamma) \cdot d(r_1, s_1)$—the difference is proportional to the length of this forced initial assignment. This immediately gives us Theorem 1—let us show this fact before we prove Lemma 3.

Proof of Theorem 1. Given any request sequence σ and an optimal matching f_σ for this sequence such that $f_\sigma(r_1) \in N(r_1)$, we can define a sequence of hybrid matchings $\{h_\sigma^t\}_{t=0}^k$, where h_σ^t is obtained by matching the first t requests $r_1, \ldots, r_t$ in σ to $f_\sigma(r_1), \ldots, f_\sigma(r_t)$ and the remaining requests $r_{t+1}, \ldots, r_k$ to $g_\sigma(r_{t+1}), \ldots, g_\sigma(r_k)$. Note that h_σ^0 is just the *Harmonic* matching g_σ, and h_σ^k produces the optimal matching f_σ. Moreover, by ignoring the servers in $\{f_\sigma(r_i) \mid i \leq t\}$ and just considering $r_{t+1}, \ldots, r_k$ as the request sequence, Lemma 3 implies

$$E[\textstyle\sum_{i=t+1}^{k} d(r_i, h_\sigma^t(r_i))] \leq E[\sum_{i=t+1}^{k} d(r_i, h_\sigma^{t+1}(r_i))] + O(\log \Gamma) \cdot d(r_{t+1}, f_\sigma(r_{t+1})),$$

since we can regard the assignment $r_{t+1} \to f_\sigma(r_{t+1}) \in N(r_{t+1})$ as the assignment $r_1 \to s_1$ used in Lemma 3. (It can be checked that the optimal assignment indeed assigns r_1 to a server in $N(r_1)$.) Now, by adding $\sum_{i=1}^{t} d(r_i, f_\sigma(r_i))$ to both sides,

$$E[\textstyle\sum_{i=1}^{k} d(r_i, h_\sigma^t(r_i))] \leq E[\sum_{i=1}^{k} d(r_i, h_\sigma^{t+1}(r_i))] + O(\log \Gamma) \cdot d(r_{t+1}, f_\sigma(r_{t+1})).$$

Summing this over all values of $t \leq k-1$, and using that $h_\sigma^0 = g_\sigma$ and $h_\sigma^k = f_\sigma$,

$$E[\textstyle\sum_{i=1}^{k} d(r_i, g_\sigma(r_i))] \leq E[\sum_{i=1}^{k} d(r_i, f_\sigma(r_i))] + O(\log \Gamma) \cdot \sum_{i=1}^{k} d(r_i, f_\sigma(r_i)).$$

The left side is the expected cost of *Harmonic*, and the right side is the cost of the optimal matching, which proves Theorem 1. ■

3.1 Proof of the Hybrid Lemma: A Coupling Argument

We now prove Lemma 3. Here is the high-level idea: recall that G is just a run of *Harmonic*, whereas H first forces $r_1 \to s_1$ (a server adjacent to r_1 on the line) and then runs *Harmonic*. So, just after r_1 has been assigned, either both G and H have the same set of free servers, or their symmetric difference is a pair of servers with no other free servers between them. (Think of the location of the

free servers in one run as being obtainable from the free servers in the other run by moving a single server without jumping over any free servers, and let δ_1 be this random distance.) Now we will couple the random runs of G and H so that this property will continue to hold (or the set of servers will become the same, after which they will proceed in lock-step). We prove that the expected difference in the costs of the two algorithms will be $O(\log \Gamma) \cdot E[\delta_1]$. Since $E[\delta_1] = d(r_1, s_1)$ by the probabilities in the *Harmonic* algorithm, Lemma 3 follows.

For convenience, we will say that the request r_t is assigned at time t, and we refer to the situation just before this assignment as being at time t^-, and the situation just after as time t^+; note that $(t-1)^+ = t^-$. Let $\mathsf{A}_G(t)$ be the set of free servers at time t^+ when running algorithm G, and $\mathsf{A}_H(t)$ be similarly defined for algorithm H. Note that if at time t^+, $\mathsf{A}_G(t) = \mathsf{A}_H(t)$, then the expected difference in costs between algorithms G and H on requests $r_{t+1}, \ldots, r_k$ is 0. Thus, we can without loss of generality only consider the time instants where $\mathsf{A}_G(t) \neq \mathsf{A}_H(t)$.

Let g_1 be the only element of $\mathsf{A}_G(1) \setminus \mathsf{A}_H(1)$ and h_1 the only element of $\mathsf{A}_H(1) \setminus \mathsf{A}_G(1)$. Let $\delta_t = d(g_t, h_t)$ be the distance between these two servers. We now give a coupling π between the executions of G and H—equivalently a coupling between the evolutions of sets $\mathsf{A}_G(t)$ and $\mathsf{A}_H(t)$—from the two different starting configurations. For a valid coupling, the marginals should give us a faithful execution of *Harmonic* on $\mathsf{A}_G(1)$ and $\mathsf{A}_H(1)$ respectively. We define a coupling π maintaining the invariant that $|\mathsf{A}_G(t) \setminus \mathsf{A}_H(t)| = 1 = |\mathsf{A}_H(t) \setminus \mathsf{A}_G(t)|$, so we need to also define the coupling only on such pairs of states. By symmetry, assume that g_1 is to the left of h_1; we will maintain the invariant that g_t lies to the left of h_t. Also, when we start, there are no available servers between g_1 and h_1, and we will also maintain the invariant that there are no available servers between g_t and h_t, so we need only define the coupling over such pairs of states.

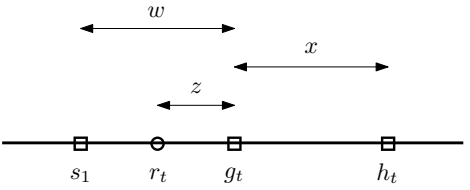
For the coupling π, we will write $\Pr_\pi[\mathcal{E}]$ to denote the probability of an event $\mathcal{E}$. This coupling also induces marginals on G and H, which we indicate by $\Pr_G[\mathcal{E}]$. We write $r \to_G s$ if G assigns r to s, and $N_G(r_t)$ will be the (at most two) neighboring free servers to the request r_t in G. (Analogous definitions hold for H.) If $r \to_G s_g$ and $r \to_H s_h$, then $\Delta c(r) := d(r, s_g) - d(r, s_h)$. Note that $\Delta c(r)$ can be negative.

Now, for the coupling π, there are four cases to consider when the request r_t arrives:

- **Case 0:** r_t's neighboring servers are identical in both G and H,
- **Case 1:** r_t lies to the left of both g_t and h_t but $g_t \in N_G(r_t)$,
- **Case 2:** r_t lies between g_t and h_t (so $g_t \in N_G(r_t)$ and $h_t \in N_H(r_t)$), and
- **Case 3:** r_t lies to the right of both g_t and h_t but $h_t \in N_H(r_t)$.

The fact that these are the only four cases follows from the invaraints we maintain in the coupling. Let us now define the coupling for these cases. (For lack of space, we defer the first and last cases to the full version.)

– For Case 1, we have the following situation:

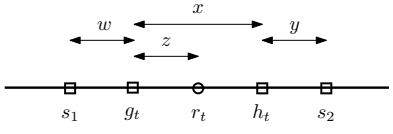


Set $p = \frac{w-z}{w}$ and $q = \frac{w-z}{w+x}$. We define the coupling π for this case:

Event	Assignments	$\Pr_\pi$	$\delta_{t+1} - \delta_t$	$\Delta c(r_t)$
1	$r_t \to_G s_1, r_t \to_H s_1$	$1-p$	0	0
2	$r_t \to_G g_t, r_t \to_H s_1$	$p-q$	w	$2z-w$
3	$r_t \to_G g_t, r_t \to_H h_t$	q	$-x$	$-x$

Note that r_t goes to s_1 with probability $1-p$ and to g_t with probability p, hence the coupling is faithful run of G. And r_t goes to s_1 with probability $1-q$ and to h_t with probability q, as it should in H.

– In Case 2, we have the following situation:



Set $p = \frac{x+y-z}{x+y}$ and $q = \frac{w+z}{w+x}$, and define the coupling as follows:

Event	Assignments	$\Pr_\pi$	$\delta_{t+1} - \delta_t$	$\Delta c(r_t)$
1	$r_t \to_G s_2, r_t \to_H h$	$1-p$	y	y
2	$r_t \to_G g, r_t \to_H h$	$p+q-1$	$-x$	$2z-x$
3	$r_t \to_G g, r_t \to_H s_1$	$1-q$	w	$-w$

Note that r_t goes to g_t with probability p and to s_2 with probability p, hence the coupling is faithful run of G. And r_t goes to s_1 with probability $1-q$ and to h_t with probability q, as it should in H.

Define $Q_{i,n}$ to be the worst-case probability of the distance between $g_{t'}$ and $h_{t'}$ eventually going above n (for some future time t'), conditioned on $d(g_t, h_t) = i$ at time t—here the worst-case is taken over all possible future request sequences, and all feasible arrangements of any number of common servers in $\mathsf{A}_G(t) \cap \mathsf{A}_H(t)$, subject to the constraint that the distance between g_t and h_t is equal to i (and where g_t is to the left of h_t and no free servers between them).

Lemma 4. *The coupling maintains the following properties:*
(i) At each step t, if $\delta_{t+1} \neq 0$, then $\Delta c(r_t) \leq \delta_{t+1} - \delta_t$. If $\delta_{t+1} = 0$, then $\Delta c(r_t) \leq \delta_t$.
(ii) $Q_{i,n} \leq i/n$.

Proof. Property (i) follows by inspection of the above tables. For the proof of Property (ii), clearly $Q_{n,n} = 1$ for all n. Now, fix some n, and suppose we know

that for all $j > \delta_t$, we have $Q_{j,n} \le j/n$. Note that for the above four cases, each of these requests at time t either makes $\delta_{t+1} = 0$ (after which it can never reach n), keeps $\delta_{t+1} = \delta_t$, or makes distance δ_{t+1} more than δ_t (upon which we can apply induction to get a bound on $Q_{\delta_t,n}$). We thus enumerate over all four possibilities:

- For case 0, the distance does not change, and so there is nothing to show.
- For case 1, we get $Q_{x,n} = (p-q)Q_{x+w,n} + (1-p)Q_{x,n}$. This gives us $Q_{x,n} = \frac{p-q}{p} Q_{x+w,n}$, and using the inductive hypothesis for $Q_{x+q,n}$, we get $Q_{x,n} \le (1-q/p)(x+w)/n = x/n$.
- For case 2, we have $Q_{x,n} = (1-p)Q_{x+y,n} + (1-q)Q_{x+w,n} \le \frac{z}{x+y}(x+y)/n + \frac{x-z}{x+w}(x+w)/n = x/n$.
- For case 3, we get $Q_{x,n} = (p-q)Q_{x+w,n} + (1-p)Q_{x,n}$. This gives us $Q_{x,n} = \frac{p-q}{p} Q_{x+w,n}$, and so $Q_{x,n} \le (1-q/p)(x+w)/n = x/n$.

In all cases, assuming that $Q_{j,n} \le j/n$ for all $j > x$, we see that $Q_{x,n} \le x/n$. This completes the proof of the lemma. ∎

We can now prove Lemma 3. We want to bound $E_G[\sum_{i=1}^k d(r_i, g_\sigma(r_i))] - E_H[\sum_{i=1}^k d(r_i, h_\sigma(r_i)]$, but since π's marginals are faithfully running G and H, we can use linearity of expectatitions to bound

$$E_\pi\left[\sum_{i=1}^k d(r_i, g_\sigma(r_i)) - \sum_{i=1}^k d(r_i, h_\sigma(r_i)\right] = E_\pi\left[\sum_{i=1}^k \Delta c(r_k)\right].$$

But by Lemma 4(i), we know that $\Delta c(r_k) \le (\delta_2 - \delta_1) + (\delta_3 - \delta_2) + \cdots + (\delta_q - \delta_{q-1}) + \delta_q$, where $\delta_{q+1} = 0$ for the first time. This is at most $2\delta_q = 2\delta_{\max}$. So it remains to bound $E_\pi[\delta_{\max}]$. We see that

$$E_\pi[\delta_{\max} \mid \delta_1] = \sum_{l=1}^{\Gamma} \Pr_\pi[\delta_{\max} \ge l | \delta_1] \le \sum_l Q_{\delta_1,l} \le \sum_{l=1}^{\Gamma} \delta_1/l = O(\log \Gamma) \cdot \delta_1$$

by Lemma 4(ii) and the definition of $Q_{j,n}$. So $E_\pi[\delta_{\max}] = O(\log k) \cdot E_\pi[\delta_1]$. Now if the two servers adjacent to r_1 were s_1 and h_1, then we have $E[\delta_1] = \frac{d(r_1,h_1)}{d(h_1,s_1)} \cdot 0 + \frac{d(r_1,s_1)}{d(h_1,s_1)} \cdot d(h_1,s_1) = d(r_1,s_1)$. This proves the hybrid lemma (Lemma 3).

4 The Random-Subtree Algorithm

We now turn to showing that a different randomized algorithm gives an $O(\log k)$ competitive ratio for the line; the proof generalizes to doubling metrics too. To start off, we use the fact that *binary* 2-HSTs approximate the line metric with $O(\log k)$ expected stretch. It is not difficult to show that the (deterministic) greedy algorithm on a binary 2-HST is $O(\log k)$-competitive compared to the optimal solution on the tree, which implies an $O(\log^2 k)$-competitive ratio in all. In this section, we show that randomization helps: a certain randomized greedy algorithm is $O(1)$-competitive on the binary 2-HST, giving us a different $O(\log k)$-competitive algorithm for the line. In fact, the proof extends to HSTs obtained from doubling metrics, and hence proves Theorem 2.

The Algorithm. Let us define the algorithm *Random-Subtree* for online metric matching on an arbitrary HST as follows: when a request r comes in, consider its lowest ancestor node a whose subtree $T(a)$ also contains a free server. Now we choose a random free server in the subtree rooted at a as follows: from among those of a's children whose subtrees contain a free server under them, we choose such a child of a uniformly at random, and repeat this process until we reach a leaf/server s—we then map r to server s. Observe that ours is a different randomized greedy algorithm from that in [9], which would have chosen a server uniformly at random from among all of the servers in $T(a)$. Our main theorem is the following.

Theorem 5. *The algorithm* Random-Subtree *is* $2(1 + 1/\epsilon)H_\Delta$*-competitive on* Δ*-ary* α*-HSTs, as long as* $\alpha \geq \max((1+\epsilon)H_\Delta, 2)$.

Since the line embeds into binary 2-HSTs with expected stretch $O(\log k)$, we get an $O(\log k)$-competitive randomized algorithm for the line. Moreover, in the full version, we show that an algorithm for Δ-ary α-HSTs satisfying the property above (with $\Delta = O(1)$) implies an algorithm for doubling metrics with an additional loss of $O(\log k)$; this proves Theorem 2.

The proof of the theorem goes thus: we first just consider the edges incident to the root (which we call root-edges) of an Δ-ary α-HST, and count the number of times these edges are used. Specifically, we show that for any sequence of requests, the number of requests that use the root-edges in our algorithm is at most H_Δ times the minimum number of requests that must use these root-edges. This "root-edges lemma" is the technical heart of our analysis; getting H_Δ instead of H_k (obtained in [9]) requires defining the right potential function, and carefully accounting for the gain we get from using the *Random-Subtree* algorithm rather than the randomized greedy algorithm of [9].

Having proved the root-edges lemma, notice that for any fixed vertex v in an HST, the subtree rooted at v is another HST on which we can apply the root-edges lemma to bound the cost incurred on the edges incident to v. Consequently, applying this for every internal vertex in the HST and summing up the results shows that the total cost remains at most $O(H_\Delta) \cdot \mathsf{Opt}$, as long as the parameter α for the HST is larger than H_Δ.

The Analysis. Consider a Δ-ary α-HST T with a set of requests $R \cup R'$ such that the requests in R originate at the leaves of T, and those in R' originate at the root. We assume that the number of servers in T is at least $|R \cup R'|$. Let $T_1, T_2, \ldots, T_\Delta$ denote the Δ child subtrees of T. Without loss of generality, we assume that T has exactly Δ child subtrees. We will use $R(T_i)$ to denote the set of requests that originate in subtree T_i. Let n_i be the number of servers in T_i, and let $M^* = \sum_{i=1}^{\Delta} \max(|R \cap R(T_i)| - n_i, 0)$. The following fact gives a lower bound for Opt.

Fact 6. *In any assignment of requests in* $R \cup R'$ *to servers, at least* $M^* + |R'|$ *requests use root-edges.*

The following crucial lemma upper-bounds the expected cost incurred by the algorithm on just the root edges.

Lemma 7 (Root-Edges Lemma). *Let the random variable M count the number of requests in $R \cup R'$ that use a root-edge when assigned by the algorithm* Random-Subtree*.*

$$E[M] \leq H_\Delta \cdot (M^* + |R'|).$$

Proof. Let the k requests $R \cup R'$ be labeled $r_1, r_2, \ldots, r_k$, where r_1 is the earliest request and r_k is the latest request. The request r_t is assigned at time t, and we refer to the situation just before this assignment as being at time t^-, and the situation just after as time t^+. Note that t^- for $t = 1$ (denoted as 1^-) represents the time before any request assignments have been made, and t^+ for $t = k$ (denoted as k^+) represents the time after all request assignments have been made. Let $R_t = \{r_t, r_{t+1}, \ldots, r_k\}$, the set of requests at time t^- that have yet to arrive. At time t^-, let $n_{i,t}$ be the number of available servers in tree T_i. A subtree T_i is said to be *open* at time t^- if $n_{i,t} > 0$ (there are available servers at time t^- in T_i). Let η_t be the number of open subtrees of T at time t^-.

Define the first $\min(n_{i,t}, |R_t \cap R(T_i)|)$ requests of T_i to be the *local requests of T_i at time t^-* (these are the ones in $R(T_i)$ that have the lowest numbered indices), and the remaining requests in T_i to be the *global requests of T_i at time t^-*.[1] Let $\mathcal{L}_{i,t}$ and $\mathcal{G}_{i,t}$ be the set of local and global requests in T_i at time t^-, and let $\mathcal{L}_t := \cup_i \mathcal{L}_{i,t}$ and $\mathcal{G}_t := \cup_i \mathcal{G}_{i,t}$. For convenience, we say that a request r_j *becomes global* at time t if r_j is local at time t^-, but r_j is global at time t^+. Let requests in $\mathcal{R}_t := R_t \cap R'$ be called *root requests of T at time t^-*.

As a sanity check, note that at the beginning (at time 1^-), the set of pending requests $R_1 = R \cup R'$, the number of pending requests in subtree T_i is $n_{i,1} = n_i$, the number of global requests in T_i is $|\mathcal{G}_{i,1}| = \max(|R \cap R(T_i)| - n_i, 0)$ (so the total number of global requests at time 1^- is M^*), and the number of root requests is $|\mathcal{R}_1| = |R'|$.

Recall that global requests of T_i must assign to servers outside of T_i: while an optimal offline algorithm can identify where to assign these global requests, an online algorithm may assign a global request from T_i to some subtree T_j that only has as many servers as future requests, which causes some local request in T_j to become global. Hence we want to upper-bound the number of future requests in R_{t+1} that become global due to our assignment for r_t. We associate with each request in R_t a "cost" at time t^- which represents this upper bound. Later, we will use the cost function to define the potential function. The cost function at time t^- is $F_t : R_t \to \mathbb{Z}_{\geq 0}$; we say it is *well-formed* if it satisfies two properties:

- $F_t(r_j) = 0$ if and only if $r_j \in \mathcal{L}_t$ (i.e., it is a local request at time t^-), and

[1] The idea behind calling requests local/global is this: assuming no servers in T_i are used up by requests from other subtrees, the local servers will be assigned within T_i by our algorithm, whereas the global ones will be assigned to other subtrees (and hence use a root-edge). Of course, as servers within T_i are used by requests in other subtrees, some local requests become global.

– for all global and root requests $r_j \in \mathcal{G}_t \cup \mathcal{R}_t$, $F_t(r_j)$ is an upper bound on the random variable η_j, the number of open subtrees at time j^-.

Constructing the Well-Formed Cost Functions. We set $F_1(r_j) = \Delta$ (the degree of the tree) for all $r_j \in \mathcal{G}_1 \cup \mathcal{R}_1$ (global and root requests at time 1^-), and $F_1(r_j) = 0$ for all $r_j \in \mathcal{L}_1$ (local requests at time 1^-). It is immediate that the map F_1 is well-formed.

Now at each time t^+, we will define the next function F_{t+1} using F_t. For this, first consider time t^-, and suppose that the map F_t is well-formed. The easy case first: If $r_t \in \mathcal{L}_t$, then define $F_{t+1}(r) = F_t(r)$ for all $r \in R_t$. In this case if a request in R_t is a local/global/root request at time t^-, it remains a local/global/root request at time t^+, so F_{t+1} is still well-formed.

On the other hand, suppose $r_t \in \mathcal{G}_t \cup \mathcal{R}_t$, i.e., it is a global or root request. Recall there are η_t open subtrees at time t^-. Each open subtree T_i contains $|R_t \cap R(T_i)|$ requests and $n_{i,t}$ free servers, so if $|R_t \cap R(T_i)| \geq n_{i,t}$ then assigning r_t to a server in this subtree would cause some request r_j in $R_t \cap R(T_i)$ to become global at time t (because $n_{i,t+1}$ would become $n_{i,t} - 1$). In this case, define $a_t(T_i) := j$, the index of the request r_j that turns global in subtree T_i. Else, if no request in $R_t \cap R(T_i)$ would become global, set $a_t(T_i) := k + i$ (which cannot be the index of any request, since there are only k requests). Let $A_t = \{a_t(T_i) \mid T_i \text{ open at time } t^-\}$; note that $|A_t| = \eta_t$. Now denote the elements of A_t by $\{p_j\}_{j=1}^{\eta_t}$ such that $p_1 < p_2 < \cdots < p_{\eta_t}$.

(Another sanity check: we claim that the last entry $p_{\eta_t} > k$; indeed, if r_t is a global or root request, there must be some open subtree T_i which has more available servers than requests.) Now, let T_i be the subtree that r_t assigns to, chosen by picking out of the open subtrees uniformly at random. We now define the map F_{t+1} at time t^+. There are two cases to consider:

– If $a_t(T_i) > k$ (i.e., none of the requests in $R(T_i) \cap R_{t+1}$ become global due to assigning r_t), then we set $F_{t+1}(r) = F_t(r)$ for all requests $r \in R_{t+1}$.
– If $a_t(T_i) \leq k$, then say $a_t(T_i) = p_{\eta_t - q + 1}$ in the ordering given above (i.e., $a_t(T_i)$ was the q^{th} largest value in A_t). Now assign $F_{t+1}(r) = F_t(r)$ for all $r \in R_{t+1} \setminus \{r_{a_t(T_i)}\}$, and $F_{t+1}(r_{a_t(T_i)}) = q - 1$.

Showing that this map F_{t+1} is well-formed is deferred to the full version. Note that maps F_t and F_{t+1} are either the same or differ on at most one request r_j that becomes global at time t, in which case $F_{t+1}(r_j)$ becomes positive. Moreover, $F_{t'}(r_j) = F_{t+1}(r_j)$ for all times $t' \in [t+1, j]$.

The Potential Function Analysis. We are now in a position to define the potential function,

$$\Phi_t = \sum_{r \in R_t} H_{F_t(r)}, \tag{4.1}$$

where we consider $H_0 = 0$. Also, define ρ_t to be the number of requests that our algorithm has already matched outside of their subtrees at time t^-. The root-edges lemma follows immediately from the following claim, proved using induction.

Lemma 8. *For all* $t \in [1, k+1]$, $E[\Phi_t + \rho_t] \leq H_\Delta \cdot (M^* + |R'|)$.

(Proof given in the full version.) Since $\rho_{k+1} = M$ and $\Phi_{k+1} = 0$, using Lemma 8 with $t = k+1$ finishes the proof of the root-edges lemma. ■

The next lemma bounds the total cost, not just the cost on the root edges, by considering every subtree in the HST and applying the root-edges lemma to each subtree.

Lemma 9. *Consider a* Δ*-ary* α*-HST* T*, any set* R *of requests at the leaves of* T*, and requests* R' *at the root of* T*, such that* $|R \cup R'|$ *is at most the number of servers in* T*. If* $\mathsf{Alg}(R \cup R', T)$ *denotes the cost of* Random-Subtree *on requests* $R \cup R'$ *on tree* T*, and* $\mathsf{Opt}(R \cup R', T)$ *the cost of the optimal solution, we have*

$$E[\mathsf{Alg}(R \cup R', T)] \leq c \cdot H_\Delta \cdot \mathsf{Opt}(R \cup R', T)$$

for $c = 2(1 + 1/\epsilon)$ *as long as* $\alpha \geq \max\{2, (1+\epsilon)H_\Delta\}$.

The lemma above directly proves Theorem 5. As an aside, note that 2-HSTs that have large degree, or binary HST's that have $\alpha \approx 1$ (say $\alpha = 1 + 1/\log k$), can both simulate star metrics, on which we have an $\Omega(\log k)$ lower bound—hence we do need some relationship between α and Δ.

References

1. Bansal, N., Buchbinder, N., Gupta, A., Naor, J.S.: An $o(\log^2 k)$-competitive algorithm for metric bipartite matching. In: Proceedings of the 15th Annual European Symposium on Algorithms, pp. 522–533 (2007)
2. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. In: STOC 2003: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, pp. 448–455 (2003)
3. Fuchs, B., Hochstattler, W., Kern, W.: Online matching on a line. Theoretical Computer Science 332, 251–264 (2005)
4. Kalyanasundaram, B., Pruhs, K.: Online weighted matching. J. Algorithms 14(3), 478–488 (1993)
5. Kalyanasundaram, B., Pruhs, K.: Online Network Optimization Problems. In: Fiat, A. (ed.) Online Algorithms 1996. LNCS, vol. 1442, pp. 268–280. Springer, Heidelberg (1998)
6. Khuller, S., Mitchell, S.G., Vazirani, V.V.: On-line algorithms for weighted bipartite matching and stable marriages. Theor. Comput. Sci. 127(2), 255–267 (1994)
7. Koutsoupias, E., Nanavati, A.: The Online Matching Problem on a Line. In: Solis-Oba, R., Jansen, K. (eds.) WAOA 2003. LNCS, vol. 2909, pp. 179–191. Springer, Heidelberg (2004)
8. Koutsoupias, E., Papadimitriou, C.H.: On the k-server conjecture. J. ACM 42, 971–983 (1995)
9. Meyerson, A., Nanavati, A., Poplawski, L.: SODA 2006: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm. In: SODA 2006: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, pp. 954–959 (2006)