A Poly-log Competitive Posted-price Algorithm for Online Metrical Matching on a Spider*

Max Bender[1], Jacob Gilbert[2], and Kirk Pruhs[1][0000−0001−5680−1753]

[1] Computer Science Deparment
University of Pittsburgh, Pittsburgh, PA 15213
mcb121@pitt.edu, kirk@pitt.edu
[2] Computer Science Department
University of Maryland
jgilber8@umd.edu

Abstract. Motivated by demand-responsive parking pricing systems we consider posted-price algorithms for the online metrical matching problem. Our main result is a polylog competitive posted-price algorithm in the case that the metric space is a spider.

1 Introduction

1.1 Motivation and Problem Statement

SFpark is San Francisco's system for managing the availability of on-street parking [3, 29, 2]. The goal of SFpark is to reduce the time and fuel wasted by drivers searching for an open parking spot. The system monitors parking usages using sensors embedded in the pavement and distributes this information in real-time to drivers via SFpark.org and phone apps. SFpark periodically adjusts parking meter pricing to manage demand, to lower prices in underutilized areas, and to raise prices in overutilized areas. Prices can range from a minimum of 25 cents to a maximum of 7 dollars per hour during normal hours, with a 18 dollars per hour cap for special events such as baseball games or street fairs. Several other cities in the world have similar demand-responsive parking pricing systems, for example Calgary has had the ParkPlus system since 2008 [1].

One natural simple model of the problem of centrally assigning drivers to parking spots to minimize time and fuel usage is the online metrical matching problem. The setting for online metrical matching consists of a collection of k servers (the parking spots) located at various locations within a metric space. The algorithm then sees an online sequence of requests over time that arrive at various locations in the metric space (the drivers arriving to look for a parking spot). In response to a request, the online algorithm must match the request (car) to some server (parking spot) that has not been previously matched. Conceptually we interpret this matching as the request (car) moving to the location of the matched server (parking spot). The objective goal is to minimize the aggregate distance traveled by the requests (cars).

In order to be implementable within the context of SFpark, algorithms for online metric matching must be posted-price algorithms. In this setting, posted-price means that before each request arrives, the algorithm sets a price on each unused server (parking spot) without knowing the location where the next request will arrive. We assume each request is a selfish agent who moves to the available server (parking spot) that minimizes the sum of the price of that server (parking spot) and the distance to that server (parking spot). The objective remains to minimize the aggregate distance traveled by the requests (cars). Thus conceptually, the objective of the parking pricing agency is minimizing social cost, not maximizing revenue.

Research into posted-price algorithms for online metrical matching was initiated in [14] as part of a broader study of the use of posted-price algorithms to minimize social cost in online optimization problems. As a posted-price algorithm is a valid online algorithm, one can not expect to obtain a better competitive ratio for posted-price algorithms than what is achievable by general online algorithms. So this research line has primarily focused on problems where the optimal competitive ratio achievable by an online algorithm is (perhaps approximately) known and seeks to determine whether a comparable competitive ratio can be (again perhaps approximately) achieved by a posted-price algorithm. The higher level goal is to determine the increase in social cost that is necessitated by the restriction that the

* Supported in part by NSF grants CCF-1421508 and CCF-1535755, and an IBM Faculty Award.

algorithm has to use posted prices to incentivize selfish agents, instead of being able to mandate agent behavior.

Before stating our results on posted-price algorithms for online metric matching we want to lay the groundwork by reviewing past work on posted-price algorithm design techniques, past work on general online algorithms for online metric matching, and past work on posted-price algorithms for online metric matching.

1.2 Past Work

The most obvious algorithmic design approach for posted-price problems is to directly design a pricing algorithm from scratch, as is done for metrical task systems in [15]. Two less direct algorithmic design paradigms have emerged in the literature. The first algorithmic design paradigm is what we will call *mimicry*. A posted-price algorithm A *mimics* an online algorithm B if the probability that B will take a particular action is equal the probability that a self-interested agent will choose this same action when the prices of actions are set using A. For example, [18] shows how to set prices to mimic the $O(1)$-competitive algorithm Slow-Fit from [8, 9] for the problem of minimizing makespan on related machines. For some problems it is not possible to mimic known competitive algorithms using posted prices. For such problems, another algorithmic design paradigm is what we will call *monotonization*. In the monotonization algorithm design approach, one first seeks to characterize the online algorithms that can be mimicked, and then design such a mimicable online algorithm (the reason for using this terminology is that in all the examples in the literature, this characterization involves some sort of monotonicity property). For example, monotonization is used in [15] to obtain an $O(k)$-competitive posted-price algorithm for the k-server problem on a line metric, in [16] to to obtain an $O(k)$-competitive posted-price algorithm for the k-server problem on a tree metric, and in [21] to obtain an $O(1)$-competitive posted-price algorithm for minimizing maximum flow time on related machines. In all of these examples, the competitive ratio achievable by the posted-price algorithm is comparable to the best competitive ratio achievable by a general online algorithm, thus showing that there is minimal increase in social cost necessitated by the use of posted-prices.

Let us now turn to known results for general algorithms for online metric matching in a general metric. There is a deterministic online algorithm that is $(2k-1)$-competitive, and no deterministic online algorithm can achieve a better competitive ratio in a star metric [22, 23]. Using HST's (Hierarchically Separated Trees) [11, 12, 17], [25] obtained an $O(\log^3 k)$-competitive randomized algorithm. The algorithm in [25] can be viewed as combining two online randomized metric matching algorithms:

HST Algorithm: An $O(\log k)$-competitive algorithm for $O(\log k)$-HST's.
Uniform Metric Space Algorithm: The natural $O(\log k)$-competitive for a uniform metric space.

Later [10] obtained an $O(\log^2 k)$-competitive randomized algorithm by replacing the HST algorithm used by [25] by a $O(\log k)$-competitive algorithm for 2-HST's.

Better competitive ratios for online metric matching can be achieved on some natural metric spaces. Let us first consider a line metric. An $O(k^{.59})$-competitive deterministic algorithm was given in [4]. Subsequently several different $O(\log k)$-competitive randomized algorithms are given in [20]. These algorithms leverage special properties of HST's constructed from a line metric. [20] also showed that the natural Harmonic algorithm is $O(\log \Delta)$-competitive, where Δ is the ratio of the distance between the furthest two servers and the distance between the closest two servers. And by applying a standard doubling approach, [20] showed how to convert this Harmonic algorithm into an $O(\log k)$-competitive algorithm. An $O(\log^2 k)$-competitive deterministic online algorithm, called RM, was given in [26], and this was later improved to $O(\log k)$ in [28]. An $\Omega(\log k)$ lower bound on the competitive ratio for certain types of natural algorithms is given in [5, 24]. A 9.001 lower bound on the competitive ratio of deterministic algorithm is given in [19]. This was improved to an $\Omega(\sqrt{\log k})$ lower bound on the competitive ratio for randomized algorithms in [27].

[26] also showed that for every metric space the competitive ratio of the RM algorithm is at most $O(\log^2 k)$ times the optimal competitive ratio achievable by a deterministic algorithm on that metric space.

We now turn to posted-price algorithms for online metric matching. [14] shows that the online algorithm Harmonic is mimicable on a line metric, and thus, using the results from [20], obtain an $O(\log \Delta)$-competitive randomized posted-price algorithm for a line metric.

[13] considered posted-price algorithms for tree metrics. [13] adopts the monotization approach, and identifies a monotonicity property that characterizes mimicable algorithms for online metrical matching

in tree metrics. This monotization property is that, as a request arrival location moves closer to a server location, the probability that the request uses that server can not decrease. While this monotization property might seem innocuous at first, standard algorithmic approaches are seemingly hopelessly non-monotone. For example, there is no hope to mimic any of the online algorithms that are based on HST's as HST's by their very nature lose too much information about the structure of a tree metric. [13] developed a type of hierarchical tree, which they call a grove, that is a refinement of an HST that retains more information about the topology of the original metric space, and showed how to approximate a tree metric by a grove in a similar way to which one can approximate a tree metric by an HST. One way to think of an HST is as a unit star where each of the leaves can be recursively thought of as a scaled-down HST. In this vein, a grove is unit tree where each of the nodes can be recursively thought of as a scaled-down grove. (Unit here means the distance of every edge is 1) One can then develop algorithms for a grove, as one does for a HST, that is, design one algorithm for the grove/HST and another algorithm for the base metric space. However, for groves the base metric is a unit tree (instead of a uniform metric space as it is for an HST). [13] gave a monotone grove algorithm that, if combined with certain types of "low-hop" monotone algorithms for a unit tree, yields a poly-log competitive monotone algorithm for a general tree metric (unfortunately its not quite a black box reduction). An algorithm for a unit tree is low-hop if the number of servers that have to move a positive distance to a parking spot is at most

$$L\left((1+\epsilon)H + \frac{\ln k}{\epsilon}\right)$$

where H is the diameter of the unit tree, $\frac{1}{\epsilon}$ is poly-log bounded, and L is the optimal/minimum number of servers that have to move a positive distance to a parking spot. The multiplicative $(1+\epsilon)$ term has to be so small because the grove algorithm is going to apply the unit tree algorithm recursively to a poly-log depth. [13] then developed a monotone low-hop algorithm for a unit tree for the online metric search problem, which is a special case of the online metric matching problem in that there is a promise that there is an optimal matching with only one nonzero length edge (most lower bounds for online metric matching are of this special type). This low-hop algorithm is based on the classic multiplicative weights algorithm in the setting of online learning from experts [6]. Conceptually there is one expert for each leaf of the tree, and this expert always recommends that the car/request travel toward this leaf. Putting this all together, the main result of [13] is a $O(\log^6 \Delta \log^2 n)$-competitive posted-price algorithm for online metric searching (not matching) on a general tree metric.

1.3 Our Contribution

Our main result is a $O(\log^5 \Delta \ \log^2 n)$-competitive posted-price algorithm for online metric matching on a spider metric. A spider is a rooted tree T in which the root ρ (sometimes called the head) is the only node of degree greater than 2. We use d to denote the degree of the root and use the term leg to refer to the leaf to root paths in the spider. We achieve our main result by designing a monotone low-hop algorithm `Spider-Match` for a unit spider, and then applying the techniques developed in [13] for groves.

As in [13], the starting point in the design of our low-hop algorithm `Spider-Match` is the classic multiplicative weights algorithm in the setting of online learning from experts [6]. However, because we consider online metric matching, instead of online metric search as in [13], we have to surmount technical difficulties that did not arise in [13]. In section 2 we give an overview of some of the key algorithmic ideas behind the design of `Spider-Match`, and an explanation of some of these technical difficulties. We strongly recommend that the reader read this section before launching into the subsequent technical sections.

In section 3 we give some preparatory notation and terminology. In section 4 we give a formal definition of the `Spider-Match` algorithm. In section 5 we give the analysis of `Spider-Match`. Finally in the appendix we show how these results can be combined with the results from [13] to obtain a $O(\log^5 \Delta \log^2 n)$-competitive posted-price algorithm for online metric matching on a spider metric (This is more or less the same as the grove analysis in [13], except for a slight refinement that cuts off a factor of $\log \Delta$).

2 Intuitive Overview On A Simple Instance

Assume in our unit spider T that each leg is of infinite length, that the root ρ contains no servers, and that each node other than ρ contains a single server. Furthermore, suppose the request sequence $r_1, r_2, \ldots$ is such that

- The first $m < d$ requests, $r_1, ..., r_m$ all arrived at the root ρ, and
- if a request r_t, $t > m$, arrives on a leg ℓ then it must arrive at the vertex on leg ℓ that is closest to the root ρ among those vertices where a request has not yet arrived.

To model this within the setting of online learning from experts [6], we assume that there are $\binom{d}{m}$ experts, one for each of the possible collection R of of m distinct legs. Initially, the expert R recommends the first m requests move down the legs in R (one request per leg) to the first available server. Subsequently each expert R recommends requests that arrive on a leg $\ell \in R$ move down to the next available server on leg ℓ, and recommends that requests that arrive on a leg $\ell \notin R$ use the server at the location where the request arrived (which exists by assumption). So each expert incurs a cost of 1 hop when a request arrives on one of its recommended legs, and a cost of 0 otherwise. (Recall that in the definition of a low-hop algorithm, all moves of nonzero distance cost one hop).

The classic multiplicative weights algorithm maintains a probability distribution π over experts R, which in turn induces a probability $\pi(s)$ that server s is available if an expert is picked according to probability distribution π. If one could design an online algorithm that also maintained the invariant that each server s is available with probability $\pi(s)$ then one could conclude that the expected number of hops used by this algorithm is at most:

$$m\left((1+\epsilon)H + \frac{\ln(d)}{\epsilon} + 1\right),$$

where H is the minimum total cost of all the experts and is thus low-hop, using the standard analysis of the multiplicative weights algorithm [6].

To maintain the invariant that each server is available with probability $\pi(s)$ we need to design a probability distribution q^r for each possible location r where the next request might arrive, where $q^r(\ell)$ is the probability that our algorithm will move a request arriving at location r to the next available server on leg ℓ. We need that probability distributions q^r to satisfy two properties:

- The probability that a server s is available will be $\pi(s)$, so the experts distribution is matched.
- These probabilities are monotone. So as r moves closer to a server s the probability that the algorithm will use server s can not decrease.

Again, at first impression the monotonicity requirement can seem innocuous, but it is actually very limiting. Our eventual design of such probability distributions q^r was by significant trial and error; we do not have a principled explanation why these are in some sense the "right" distributions.

An even bigger technical challenge arises when we relax the requirement that the m requests at the root arrive at the start, and instead may arrive anywhere in the sequence. Then the algorithm's estimate m of the number of root requests, and thus the number $\binom{d}{m}$ of experts, can increase when a request arrives at the root. This request then yields in a new probability distribution $\tilde{\pi}$ over $\binom{d}{m+1}$ experts that replaces the prior probability distribution π over $\binom{d}{m}$ experts. First, we need to show how to adapt the standard analysis of the multiplicative weights algorithm to handle this. Then we need to design a probability distribution $\tilde{q}$ where $\tilde{q}(\ell)$ gives the probability that this request at the root will move to the first available server on leg ℓ. We need the $\tilde{q}$ to satisfy the following two properties:

- If the next request arrives at the root then the probability that a server s is available will be $\tilde{\pi}(s)$, so the new experts distribution is matched.
- The probability distributions q^r and $\tilde{q}$ are monotone. So as r moves closer to a server s the probability that the algorithm will use server s can not decrease.

As the probability distributions q^r and $\tilde{q}$ are designed to match different experts distributions, it is challenging to attain monotonicity. Again our design of $\tilde{q}$ derived from significant trial and error.

This is illustrative of a general phenomenon, it is difficult to maintain monotonicity for any sort of algorithmic design that has the form:

If a property P is true about a new request then take action A, else take action B.

when some request locations will make property P true and some requests locations will make property P false. In this case, the actions A and B have to be carefully coordinated with each other, and with the locations where property P is true, to maintain monotonicity.

Roughly speaking, on a general instance, our algorithm `Spider-Match` works as follows. If there is an available server between the current request and the root then the request moves to the first available

server on the path to the root. Intuitively, these are "easy" requests that we account for without having to invoke multiplicative weights. Otherwise if there is a hole between the current request and the root, then the request is matched to the available server on leg ℓ that is closest to the root with a probability $q(\ell)$. Roughly speaking, a hole is a server location that does not contain an available server, but that could have contained an available server if the prior random events internal to the algorithm had been different. Otherwise, the request is matched to the available server on leg ℓ that is closest to the root with a probability $\tilde{q}(\ell)$.

3 Notation and Terminology

Definition 1. – A spider metric is a rooted tree metric $(T = ((V, \rho), E), x)$, an acyclic connected graph consisting of vertices V and edges E with a particular vertex ρ marked as the root, along with a distance metric $x : V \times V \to \mathbb{R}$ satisfying
  i) $x(u, v) = 0$ if and only if $u = v$,
  ii) $x(u, v) = x(v, u)$, and
  iii) $x(u, v) \leq x(u, w) + x(w, v)$
  for all $u, v, w \in V$.
- The degree d of a tree is the degree of the root ρ.
- A spider metric is a rooted tree metric $(T = ((V, \rho), E), x)$ where ρ is the single vertex of degree greater than 2.
- A server s is a leaf-server if there are no other servers in the subtree rooted at s.
- Let $L(T) = \{\mu_1, ..., \mu_d\}$ denote the collection of leaf-servers.
- For $\ell \in [d]$, define $T_\ell \subseteq V$ as the set of servers on the path from the root ρ to μ_ℓ, inclusive. T_ℓ is referred to as the ℓ^{th} leg of T.
- A server s on T_ℓ becomes a *hole* when either
  a) a request r_t arrives on T_λ where $\lambda \neq \ell$ and r_t is matched to s,
  b) or a request r_t arrives on the path from ρ to s (inclusive of ρ, non-inclusive of s) and matches to s.
- A hole s is *filled* and loses its status as a hole when a request r_t arrives such that
  a) s is on the path from r_t to ρ, inclusive,
  b) and there is no available server on the path from r_t to s.
- The number of holes at time t is denoted by m_t.
- We define T_ℓ to be alive if there is still an available server or a hole in T_ℓ and dead otherwise.
- Let $\mathcal{A}^t = \{\ell \in [d] \mid T_\ell \text{ is alive just before the arrival of } r_t\}$.
- If $r_t \neq \rho$, let $\ell_t = \ell$ such that $r_t \in T_\ell$. Otherwise, if $r_t = \rho$, let $\ell_t = 0$.
- Let χ_t denote the sum of available servers and holes on the path from r_t to the root ρ, inclusive.

4 The Spider-Match Algorithm Design

We formally define the `Spider-Match` algorithm on a given rooted spider T with root ρ. The probability distributions $q^t_\sigma(\ell)$ and $\tilde{q}^t_\sigma(\ell)$, used in the algorithm description, are defined after the algorithm (as this seems more natural).

Definition 2 (`Spider-Match` Algorithm).
`Spider-Match` maintains an internal setting σ, initialized to $\emptyset$, representing which legs of the spider `Spider-Match` currently has holes on. The algorithm operates in two phases, starting with the core phase and possibly transitioning to the epilogue. During the **core phase**, at the arrival of r_t:

1. If $\chi_t > 1$, match r_t to the closest available server on the path from r_t to the root, inclusive (as we shall see, in this case there must be at least one such available server).
2. If $\chi_t = 1$:
   (a) If there is an available server on the path from r_t to ρ, match r_t to that server.
   (b) Otherwise, if r_t does not kill ℓ_t, then r_t is matched to the first available server in T_ℓ with probability $q^t_\sigma(\ell)$. In this case, $\sigma \leftarrow \sigma \setminus \{\ell_t\} \cup \{\ell\}$.
   (c) Otherwise, r_t is matched to the first available server in T_ℓ with probability $\tilde{q}^t_\sigma(\ell)$. In this case, $\sigma \leftarrow \sigma \setminus \{\ell_t\} \cup \{\ell\}$.
   In this case, we say that r_t was *collocated* if the server s that contributed to χ_t was collocated with r_t; otherwise, r_t is *non-collocated*.

3. If $\chi_t = 0$, r_t is matched to the first available server in T_ℓ with probability $\tilde{q}^t_\sigma(\ell)$. In this case, $\sigma \leftarrow \sigma \cup \{\ell\}$. If $m_{t+1} = |\mathcal{A}^{t+1}|$, the epilogue immediately begins.

In the **epilogue phase**, at the arrival of r_t:

1. If there is an available server on the path from r_t to ρ, match r_t to the closest available server on the path from r_t to the root, inclusive.
2. Else if $r_t = \rho$ or T_{ℓ_t} is dead, match r_t to the first available server on T_ℓ where ℓ is chosen from $\mathcal{A}^t$ uniformly at random.
3. Else match r_t to the first available server in T_{ℓ_t}.

In order to define q and $\tilde{q}$, we let

$$\begin{aligned} n^t_\ell &= |\{r_i \mid i \le t,\ r_i \in T_\ell,\ \text{and } \chi_i = 1\}|, \\ w^t_\ell &= (1-\epsilon)^{n^t_\ell} \text{ for } \ell \in [d], \\ w^t_R &= \prod_{\ell \in R} w^t_\ell = (1-\epsilon)^{\sum_{\ell \in R} n^t_\ell} \text{ for } R \in 2^{[d]}, \\ W^t(\mathcal{X}) &= \sum_{R \in \mathcal{X}} w^t_R \text{ for } \mathcal{X} \in 2^{2^{[d]}}. \end{aligned}$$

This lets us now define q and $\tilde{q}$ as follows:

$$q^t_\sigma(\ell) = \begin{cases} 0 & \text{if } \ell \in (\sigma \setminus \{\ell_t\}) \cup ([d] \setminus \mathcal{A}_t) \\ \dfrac{\epsilon w^t_\ell \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell_t, \ell\}}{m_t - 1}} \frac{w^t_T}{m_t - |\sigma \cap T|}}{(1-\epsilon) w^t_{\ell_t} W^t\left(\binom{\mathcal{A}^t \setminus \{\ell_t\}}{m_t - 1}\right) + W^t\left(\binom{\mathcal{A}^t \setminus \{\ell_t\}}{m_t}\right)} & \text{if } \ell \in \mathcal{A}^t \setminus \sigma \\ 1 - \sum_{\lambda \neq \ell_t} q^t_\sigma(\lambda) & \text{if } \ell = \ell_t \end{cases}$$

$$\tilde{q}^t_\sigma(\ell) = \begin{cases} 0 & \text{if } \ell \in \sigma \cup ([d] \setminus \mathcal{A}^t) \\ \dfrac{w^t_\ell \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell\}}{m_t}} \frac{w^t_T}{m_t + 1 - |\sigma \cap T|}}{W^t\left(\binom{\mathcal{A}^t}{m_t + 1}\right)} & \text{if } \ell \in \mathcal{A}^t \setminus \sigma \end{cases}$$

5 Analysis of the Spider-Match Algorithm

In this section, we first show in Lemma 2 that `Spider-Match` is well defined. We then show in Lemma 3 that `Spider-Match` follows the multiplicative updates method experts' distribution to maintain its holes. In Lemma 4 we show that `Spider-Match` is monotone. Lastly, we show the main theorem 1 of this section, where we bound the number of times `Spider-Match` pays to handle requests.

Lemma 1. By construction of `Spider-Match`, it follows that until the epilogue there is at most one hole per subtree T_ℓ.

Proof. This follows from the fact that `Spider-Match` either matches a request to an available server and thus does not make a new hole, or creates a new hole on a leg without a hole since by the definitions of q and $\tilde{q}$, there is 0 probability of matching to a leg in $\sigma \setminus \{\ell_t\}$. Note that there will always be a leg without a hole for a request to be to matched to in the core phase, else we move to the epilogue phase.

Lemma 2. `Spider-Match` is well defined.

Proof. We will show that

a) $\sum_{\ell \in [d]} q^t_R(\ell) = 1$,
b) $\sum_{\ell \in [d]} \tilde{q}^t_R(\ell) = 1$

for any $R \in 2^{[d]}$.

For a), note that $q^t_R(\ell_t) = 1 - \sum_{\lambda \neq \ell_t} q^t_R(\lambda)$, so it suffices to show that $q^t_R(\ell) \le 1$ for all ℓ. Since all terms in the denominator are positive, $\epsilon < 1$, and $\frac{1}{m_t - |R \cap T|} \le 1$, it follows that $q^t_R(\ell) \le 1$ for all ℓ.

For b), since $\tilde{q}_R^t(\ell) = 0$ for $\ell \in R \cup ([d] \setminus \mathcal{A}^t)$, it remains to show that $\sum_{\ell \in \mathcal{A}^t \setminus R} \tilde{q}_R^t(\ell) = 1$.

$$\begin{aligned}
\sum_{\ell \in \mathcal{A}^t \setminus R} \tilde{q}_R^t(\ell) &= \frac{\sum_{\ell \in \mathcal{A}^t \setminus R} w_\ell^t \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell\}}{m_t}} \frac{w_T^t}{m_t + 1 - |R \cap T|}}{W^t\left(\binom{\mathcal{A}^t}{m_t+1}\right)} && \text{by Def. 2} \\
&= \frac{\sum_{\ell \in \mathcal{A}^t \setminus R} \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell\}}{m_t}} \frac{w_{T \cup \{\ell\}}^t}{m_t + 1 - |R \cap (T \cup \{\ell\})|}}{W^t\left(\binom{\mathcal{A}^t}{m_t+1}\right)} && \text{since weights are multiplicative and } \ell \notin R \\
&= \frac{\sum_{T \in \binom{\mathcal{A}^t}{m_t+1}} \frac{|T \setminus R|}{m_t + 1 - |R \cap T|} w_T^t}{W^t\left(\binom{\mathcal{A}^t}{m_t+1}\right)} \\
&= \frac{\sum_{T \in \binom{\mathcal{A}^t}{m_t+1}} w_T^t}{W^t\left(\binom{\mathcal{A}^t}{m_t+1}\right)} && \text{as } |T \setminus R| = |T| - |T \cap R| \\
& && = m_t + 1 - |T \cap R| \text{ since} \\
& && T \in \binom{\mathcal{A}_t}{m_t + 1} \\
&= \frac{W^t\left(\binom{\mathcal{A}^t}{m_t+1}\right)}{W^t\left(\binom{\mathcal{A}^t}{m_t+1}\right)} = 1 && \text{by Def. 2}
\end{aligned}$$

Definition 3 (p_T^t and π_T^t). We denote p_T^t as the probability that the internal parameter σ of the algorithm is $T \in \binom{\mathcal{A}^t}{m_t}$ just before the arrival of r_t. We define

$$\pi_T^t = \frac{w_T^t}{W^t\left(\binom{[d]}{m_t}\right)}$$

Lemma 3. For any $R \in \binom{[d]}{m_t}$, we have $p_R^t \geq \pi_R^t$.

Proof. To conserve space, this proof has been moved to the appendix.

We next show that this algorithm is monotone and hence induces a pricing scheme as shown in [13].

Lemma 4. `Spider-Match` is monotone.

Proof. Let $r_t \to_{\texttt{Spider-Match}} s$ denote the event that r_t is matched by `Spider-Match` to s. We must show that $\Pr[r_t \to_{\texttt{Spider-Match}} s \mid r_t = u] \leq \Pr[r_t \to_{\texttt{Spider-Match}} s \mid r_t = v]$ for all $u, v \in V$ and $s \in S$ where v is on the path from u to s. Note that u and v can belong to separate subtrees; and if $s = v$, then $\Pr[r_t \to_{\texttt{Spider-Match}} s \mid r_t = v] = 1$, so we can assume that s is not collocated with v. The claim is also trivial for the case where $u = v$. Letting $\chi_t^v = \chi_t \mid r_t = v$, we break the proof into the following cases:

a. $\chi_t^u = 0$ and
   i) $\chi_t^v = 0$
   ii) $\chi_t^v = 1$
   iii) $\chi_t^v > 1$
b. $\chi_t^u > 1$ and
   i) $\chi_t^v = 0$
   ii) $\chi_t^v = 1$
   iii) $\chi_t^v > 1$
c. $\chi_t^u = 1$ and
   i) $\chi_t^v > 1$
   ii) $\chi_t^v = 1$
   iii) $\chi_t^v = 0$

Throughout this proof, we will define ℓ_v such that v is on T_{ℓ_v}.

For a.i), we have that $\Pr[r_t \to_{\texttt{Spider-Match}} s \mid r_t = u] = \Pr[r_t \to_{\texttt{Spider-Match}} s \mid r_t = v]$, so the claim is trivially true.

For a.ii), note that if $\ell_v \in \sigma$ then $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$. However, if $\ell_v \notin \sigma$, then there is an available server in between u and v, so $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$.

For a.iii), note that there must be some available server between u and v, as $\chi_t^v > 1$; thus we have that $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$.

For b.i) and b.ii), note that there must be some available server between u and v, as $\chi_t^u > 1 \geq \chi_t^v$; thus we have that $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$.

For b.iii), if u and v are on T_ℓ and T_λ respectively, where $\ell \neq \lambda$, then there are available servers between u and v and thus $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$. If u and v are on the same T_ℓ and if s is the closest available server to u on the path from u to ρ, then $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = \Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = v] = 1$. Otherwise, $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$.

For c.i), note that there must be some available server between u and v, as $\chi_t^v > 1$; thus we have that $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$.

For c.ii), first if $\ell_u = \ell_v$ then $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = \Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = v]$. Otherwise, note that if $\ell_v \in \sigma$ then $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$. However, if $\ell_v \notin \sigma$, then there is an available server in between u and v, so $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$.

For c.iii), first if $\ell_u \notin \sigma$ then $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = 0$. Otherwise, if $\ell_t = \ell_u$ would kill ℓ_u, then $\Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = u] = \Pr[r_t \rightarrow_{\texttt{Spider-Match}} s \mid r_t = v]$. Otherwise, we must show that $q_\sigma^t(\ell) \leq \tilde{q}_\sigma^t(\ell)$. Since $\epsilon \leq \frac{1}{3}$ we note

$$q_\sigma^t(\ell) = \frac{\epsilon w_\ell^t \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell_t, \ell\}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|}}{(1-\epsilon) w_{\ell_t}^t W^t\left(\binom{\mathcal{A}^t \setminus \{\ell_t\}}{m_t - 1}\right) + W^t\left(\binom{\mathcal{A}^t \setminus \{\ell_t\}}{m_t}\right)} \leq \frac{\epsilon}{1-\epsilon} \frac{w_\ell^t \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell_t, \ell\}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|}}{w_{\ell_t}^t W^t\left(\binom{\mathcal{A}^t \setminus \{\ell_t\}}{m_t - 1}\right) + W^t\left(\binom{\mathcal{A}^t \setminus \{\ell_t\}}{m_t}\right)}$$

$$= \frac{\epsilon}{1-\epsilon} \frac{w_\ell^t \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell_t, \ell\}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|}}{W^t\left(\binom{\mathcal{A}^t}{m_t}\right)} \leq \frac{1}{2} \frac{w_\ell^t \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell_t, \ell\}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|}}{W^t\left(\binom{\mathcal{A}^t}{m_t}\right)}. \tag{1}$$

Hence, it suffices to prove that $\frac{w_\ell^t \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell_t, \ell\}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|}}{W^t\left(\binom{\mathcal{A}^t}{m_t}\right)} \leq 2\tilde{q}_\sigma^t(\ell)$. By cross multiplication, this becomes:

$$W^t\left(\binom{\mathcal{A}^t}{m_t + 1}\right) \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell_t, \ell\}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|} \leq 2W^t\left(\binom{\mathcal{A}^t}{m_t}\right) \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell\}}{m_t}} \frac{w_T^t}{m_t + 1 - |\sigma \cap T|} \tag{2}$$

Letting $\mathcal{P} = \mathcal{A}^t \setminus \{\ell_t, \ell\}$, we note

$$W^t\left(\binom{\mathcal{A}^t}{m_t + 1}\right) = w_{\{\ell_t, \ell\}}^t W^t\left(\binom{\mathcal{P}}{m_t - 1}\right) + \left(w_{\ell_t}^t + w_\ell^t\right) W^t\left(\binom{\mathcal{P}}{m_t}\right) + W^t\left(\binom{\mathcal{P}}{m_t + 1}\right), \tag{3}$$

$$W^t\left(\binom{\mathcal{A}^t}{m_t}\right) = w_{\{\ell_t, \ell\}}^t W^t\left(\binom{\mathcal{P}}{m_t - 2}\right) + \left(w_{\ell_t}^t + w_\ell^t\right) W^t\left(\binom{\mathcal{P}}{m_t - 1}\right) + W^t\left(\binom{\mathcal{P}}{m_t}\right), \tag{4}$$

$$\text{and} \quad \sum_{T \in \binom{\mathcal{A}^t \setminus \{\ell\}}{m_t}} \frac{w_T^t}{m_t + 1 - |\sigma \cap T|} = w_{\ell_t}^t \sum_{T \in \binom{\mathcal{P}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|} + \sum_{T \in \binom{\mathcal{P}}{m_t}} \frac{w_T^t}{m_t + 1 - |\sigma \cap T|} \tag{5}$$

We will break down the left-hand side of Eq. 2 into the following disjoint inequalities, where the right-hand sides of the following inequalities are all disjoint sums from the right-hand side of Eq. 2. Hence to show Eq. 2 it suffices to show the following three inequalities:

1) $W^t\left(\binom{\mathcal{P}}{m_t - 1}\right) \sum_{T \in \binom{\mathcal{P}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|}$

$\leq 2W^t\left(\binom{\mathcal{P}}{m_t - 2}\right) \sum_{T \in \binom{\mathcal{P}}{m_t}} \frac{w_T^t}{m_t + 1 - |\sigma \cap T|} + 2W^t\left(\binom{\mathcal{P}}{m_t - 1}\right) \sum_{T \in \binom{\mathcal{P}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|}$

2) $W^t\left(\binom{\mathcal{P}}{m_t}\right) \sum_{T \in \binom{\mathcal{P}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|} \leq 2W^t\left(\binom{\mathcal{P}}{m_t - 1}\right) \sum_{T \in \binom{\mathcal{P}}{m_t}} \frac{w_T^t}{m_t + 1 - |\sigma \cap T|}$

3) $W^t\left(\binom{\mathcal{P}}{m_t + 1}\right) \sum_{T \in \binom{\mathcal{P}}{m_t - 1}} \frac{w_T^t}{m_t - |\sigma \cap T|} \leq 2W^t\left(\binom{\mathcal{P}}{m_t}\right) \sum_{T \in \binom{\mathcal{P}}{m_t}} \frac{w_T^t}{m_t + 1 - |\sigma \cap T|}$

Note that the first one is trivially true. For 2 and 3, note that we are effectively taking a summation on both sides over weighted multisets. In order to show the result, we will try to match terms on the left hand side to terms at least as big on the right hand side.

Call a function $f : A \times B \to C \times D$ *useful* if f satisfies:

1. f is an injection
2. if $f(a,b) = (c,d)$, then $c = a \setminus \{\mu\}$ and $d = b \cup \{\mu\}$ for some $\mu \in a \setminus b$.

Suppose $f : \binom{\mathcal{P}}{m_t} \times \binom{\mathcal{P}}{m_t-1} \to \binom{\mathcal{P}}{m_t-1} \times \binom{\mathcal{P}}{m_t}$ is useful. Let $g : \binom{\mathcal{P}}{m_t} \times \binom{\mathcal{P}}{m_t-1} \to \binom{\mathcal{P}}{1}$ be defined such that $f(a,b) = (a \setminus g(a,b),\ b \cup g(a,b))$. Then

$$\begin{aligned}
W^t\left(\binom{\mathcal{P}}{m_t}\right) \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_T^t}{m_t - |\sigma \cap T|} &= \sum_{R \in \binom{\mathcal{P}}{m_t}} \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_R^t w_T^t}{m_t - |\sigma \cap T|} \\
&\leq 2 \sum_{R \in \binom{\mathcal{P}}{m_t}} \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_R^t w_T^t}{m_t + 1 - |\sigma \cap T|} \\
&= 2 \sum_{R \in \binom{\mathcal{P}}{m_t}} \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_{R \setminus g(R,T)}^t w_{T \cup g(R,T)}^t}{m_t + 1 - |\sigma \cap T|} \\
&\leq 2 \sum_{R \in \binom{\mathcal{P}}{m_t}} \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_{R \setminus g(R,T)}^t w_{T \cup g(R,T)}^t}{m_t + 1 - |\sigma \cap (T \cup g(R,T))|} \\
&\leq 2 \sum_{R \in \binom{\mathcal{P}}{m_t-1}} \sum_{T \in \binom{\mathcal{P}}{m_t}} \frac{w_R^t w_T^t}{m_t + 1 - |\sigma \cap T|} \\
&= 2W^t\left(\binom{\mathcal{P}}{m_t - 1}\right) \sum_{T \in \binom{\mathcal{P}}{m_t}} \frac{w_T^t}{m_t + 1 - |\sigma \cap T|}
\end{aligned}$$

Similarly, suppose $f : \binom{\mathcal{P}}{m_t+1} \times \binom{\mathcal{P}}{m_t-1} \to \binom{\mathcal{P}}{m_t} \times \binom{\mathcal{P}}{m_t}$ is useful. Let $g : \binom{\mathcal{P}}{m_t+1} \times \binom{\mathcal{P}}{m_t-1} \to \binom{\mathcal{P}}{1}$ be defined such that $f(a,b) = (a \setminus g(a,b),\ b \cup g(a,b))$. Then

$$\begin{aligned}
W^t\left(\binom{\mathcal{P}}{m_t + 1}\right) \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_T^t}{m_t - |\sigma \cap T|} &= \sum_{R \in \binom{\mathcal{P}}{m_t+1}} \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_R^t w_T^t}{m_t - |\sigma \cap T|} \\
&\leq 2 \sum_{R \in \binom{\mathcal{P}}{m_t+1}} \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_R^t w_T^t}{m_t + 1 - |\sigma \cap T|} \\
&= 2 \sum_{R \in \binom{\mathcal{P}}{m_t+1}} \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_{R \setminus g(R,T)}^t w_{T \cup g(R,T)}^t}{m_t + 1 - |\sigma \cap T|} \\
&\leq 2 \sum_{R \in \binom{\mathcal{P}}{m_t+1}} \sum_{T \in \binom{\mathcal{P}}{m_t-1}} \frac{w_{R \setminus g(R,T)}^t w_{T \cup g(R,T)}^t}{m_t + 1 - |\sigma \cap (T \cup g(R,T))|} \\
&\leq 2 \sum_{R \in \binom{\mathcal{P}}{m_t}} \sum_{T \in \binom{\mathcal{P}}{m_t}} \frac{w_R^t w_T^t}{m_t + 1 - |\sigma \cap T|} \\
&\leq 2W^t\left(\binom{\mathcal{P}}{m_t}\right) \sum_{T \in \binom{\mathcal{P}}{m_t}} \frac{w_T^t}{m_t + 1 - |\sigma \cap T|}
\end{aligned}$$

Hence if these useful functions exist, then the proof is complete.

We now turn to showing the existence of these functions.

1. A useful function $f : \binom{\mathcal{P}}{m_t} \times \binom{\mathcal{P}}{m_t-1} \to \binom{\mathcal{P}}{m_t-1} \times \binom{\mathcal{P}}{m_t}$ exists.
2. A useful function $f : \binom{\mathcal{P}}{m_t+1} \times \binom{\mathcal{P}}{m_t-1} \to \binom{\mathcal{P}}{m_t} \times \binom{\mathcal{P}}{m_t}$ exists.

For the first statement, consider a bipartite graph $G = ((X,Y),E)$ where $X = \binom{\mathcal{P}}{m_t} \times \binom{\mathcal{P}}{m_t-1}$, $Y = \binom{\mathcal{P}}{m_t-1} \times \binom{\mathcal{P}}{m_t}$, and $\{(R_1,T_1),(R_2,T_2)\} \in E$ if $R_2 = R_1 \setminus \{\ell\}$ and $T_2 = T_1 \cup \{\ell\}$ for some $\ell \in R_1 \setminus T_1$. Note that by this choice of edges, the graph is divided into disjoint unions where for each maximal connected subgraph G_Q there exists some multi set Q so that for each element (R,T) of the subgraph G_Q we have that $Q = R \uplus T$. Furthermore, this subgraph is $m_t - |\Delta^Q|$-regular, where $\Delta^Q = \{x \in Q \mid n_x^Q > 1\}$, so by Hall's theorem there exists a perfect matching. Since there exists a perfect matching on each maximal connected subgraph, there is a perfect matching between the two sets. The function induced by this matching is useful by construction.

For the second statement, consider a bipartite graph $G = (X,Y),E)$ where $X = \binom{\mathcal{P}}{m_t+1} \times \binom{\mathcal{P}}{m_t-1}$, $Y = \binom{\mathcal{P}}{m_t} \times \binom{\mathcal{P}}{m_t}$, and $\{(R_1,T_1),(R_2,T_2)\} \in E$ if $R_2 = R_1 \setminus \{\ell\}$ and $T_2 = T_1 \cup \{\ell\}$ for some $\ell \in R_1 \setminus T_1$. Note that by this choice of edges, the graph is divided into disjoint unions where for each maximal connected subgraph $G_Q = ((X_Q,Y_Q),E_Q)$ there exists some multi set Q so that for each element (R,T) of the subgraph G_Q we have that $Q = R \uplus T$. Furthermore, this subgraph is $(m_t + 1 - |\Delta^Q|, m_t - |\Delta^Q|)$-regular, where $\Delta^Q = \{x \in Q \mid n_x^Q > 1\}$, so by Hall's theorem there exists a matching saturating X_Q. Thus, there is a matching saturating X. The function induced by this matching is useful by construction.

Lastly, we show the main theorem of this section, where we bound the number of times we pay a positive cost as a function of the height H of our tree and the degree d of the root. For the analysis of our algorithm, we will consider a metric search problem which we think will help the reader understand the analysis of Theorem 1. In this metric search problem, the algorithm sees a set of available parking spots located within a metric space. Over time, two types of events can occur: a car can enter the space, or a parking spot can be decommissioned. The algorithm's job is to always keep cars matched to available parking spaces, where each space can be filled by just one car. When a car arrives, it must be moved to an available parking space; when a parking space is decommissioned, if a car had been parked at it, then that car must be moved to a different available parking space. Clearly, any request sequence can be simulated by equating the servers to parking spaces and requests to cars. Alternatively, if a request arrives at a server for which no other request has arrived at, this can equivalently be thought of as decommissioning that parking spot. As such, given $\boldsymbol{r}$, we will create a sequence $\boldsymbol{\eta}$ of events for the parking problem described above according to the following deterministic function:

Definition 4 (The η sequence). Given r_t, define η_t by:

1. If $\chi_t = 0$: η_t is the event that a new car enters the space at the location of r_t.
2. If $\chi_t \geq 1$:
   (a) if r_t is collocated with an available server or if r_t is collocated with a parked car which has already moved: η_t is the event that the parking spot at r_t is decommissioned.
   (b) else: η_t is the event that a new car enters the space at the location of r_t.

Let L_t denote the number of cars in the system after the η_t event. Then, any optimal matching solution must contain at least L_t positive cost matchings to handle the requests $\{r_1, ..., r_t\}$. Furthermore, it follows from the definition of m_t that $m_t \leq L_t$.

Theorem 1.

$$\boldsymbol{E}\left[\sum_{t=1}^{n} \mathbf{1}^r(t)\right] \leq L_t\left((1+\epsilon)H + \frac{\ln d}{\epsilon} + 1\right)$$

where $\mathbf{1}^{\boldsymbol{r}}(t)$ is an indicator random variable that is 1 if `Spider-Match` *pays positive cost to match r_t and* 0 *otherwise.*

Proof. Our proof relies on the following claim:

Claim. Let r_τ denote the first request such that $\chi_t = 0$ and let κ denote the number of requests r_t such that $\chi_t = 0$. First, let $\boldsymbol{r}'$ be the request sequence $\boldsymbol{r}$ after removing any request for which $\chi_t = 0$. Now consider the alternate request sequence $\boldsymbol{r}''$ defined such that $r''_t = r_t$ for $t < \tau$, $r''_t = \rho$ for $\tau \leq t < \tau + \kappa$, and $r''_{\tau+\kappa+t} = r'_{\tau+t}$ for $t \geq 0$. Then $\mathbf{E}\left[\sum_{t=1}^{n} \mathbf{1}^{\boldsymbol{r}}(t)\right] \leq \mathbf{E}\left[\sum_{t=1}^{n} \mathbf{1}^{\boldsymbol{r}''}(t)\right]$.

First, we show that with this claim the result follows: because of the claim, we can assume $r = r''$, so that for $t \notin \{\tau, ..., \tau + \kappa - 1\}$ it follows that $\chi_t \neq 0$. We define cost vectors

$$c_R^t = \begin{cases} 1 & \ell_t \in R \text{ or } R \cap \mathcal{D}^t \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathcal{D}^t$ is the set of dead legs at time t.

Define δ_R^t such that

$$p_R^t = \pi_R^t + \delta_R^t \tag{6}$$

for all $R \in \binom{[d]}{m_t}$. By Lemma 3, it follows that $\delta_R^t \geq 0$ for all R. Then we have that

$$\sum_{R \in \binom{\mathcal{A}^t}{m_t}} p_R^t = \sum_{R \in \binom{\mathcal{A}^t}{m_t}} \pi_R^t + \sum_{R \in \binom{\mathcal{A}^t}{m_t}} \delta_R^t, \tag{7}$$

where $\sum_{R \in \binom{\mathcal{A}^t}{m_t}} p_R^t = 1$ by construction.

Then, if $\chi_t = 1$ and η_t is a decommissioning of a parking spot, we have

$$\begin{aligned}
\mathbf{E}\left[\mathbf{1}^{\boldsymbol{r}}(t)\right] &= \sum_{\substack{R \in \binom{\mathcal{A}^t}{m_t} \\ \text{s.t. } \ell_t \in R}} p_R^t \\
&= \sum_{\substack{R \in \binom{\mathcal{A}^t}{m_t} \\ \text{s.t. } \ell_t \in R}} \delta_R^t + \sum_{\substack{R \in \binom{\mathcal{A}^t}{m_t} \\ \text{s.t. } \ell_t \in R}} \pi_R^t && \text{definition of } \delta_R^t \\
&\leq \sum_{R \in \binom{\mathcal{A}^t}{m_t}} \delta_R^t + \sum_{\substack{R \in \binom{\mathcal{A}^t}{m_t} \\ \text{s.t. } \ell_t \in R}} \pi_R^t \\
&= \sum_{R \cap \mathcal{D}^t \neq \emptyset} \pi_R^t + \sum_{\substack{R \in \binom{\mathcal{A}^t}{m_t} \\ \text{s.t. } \ell_t \in R}} \pi_R^t && \text{definition of } \delta_R^t \\
&= \vec{c}^{\,\boldsymbol{t}} \cdot \vec{\pi}^t && \text{definition of } \vec{c}^{\,\boldsymbol{t}}.
\end{aligned}$$

Let $\mathcal{T} = \{t \mid \eta_t \text{ is a decommissioning}\}$.

The Multiplicative Weights guarantee from [7] gives us that

$$\forall R \in \binom{[d]}{m_t}, \ \sum_{t \in \mathcal{T}} \vec{c}^{\,\boldsymbol{t}} \cdot \vec{\pi}^t \leq (1+\epsilon) \sum_{t \in \mathcal{T}} c_R^t + \frac{\ln \binom{[d]}{m_t}}{\epsilon}$$

Let us choose R to be the last m_t legs that die; the result follows.

Lastly, to see that the claim holds, let $\tilde{\tau}$ denote the last time t such that $\chi_t = 0$. Then for $t \notin \{\tau, ..., \tilde{\tau}\}$, it follows that $\mathbf{E}\left[\mathbf{1}^{\boldsymbol{r}}(t)\right] = \mathbf{E}\left[\mathbf{1}^{\boldsymbol{r}''}(t)\right]$. Next, for any $t \in \{\tau, ..., \tilde{\tau}\}$, let $r_{t'}$ denote the request corresponding to r_t''. Then the number of holes at time t under $\boldsymbol{r}''$ is at least the number of holes at time t' under $\boldsymbol{r}$ by construction of r'', thus $\mathbf{E}\left[\mathbf{1}^{\boldsymbol{r}}(t)\right] \geq \mathbf{E}\left[\mathbf{1}^{\boldsymbol{r}''}(t)\right]$. The claim follows, completing the proof.

6 Conclusion

The obvious immediate open question is whether a poly-log competitive posted-price algorithm exists for online metric matching on a general tree metric. The most immediate problem that one runs into when trying to apply the approach of [13], and that we apply here, is that it is not at all clear what the "right" choice of experts is. Each of the natural choices has fundamental issues that seem challenging to overcome.

Acknowledgements: We thank Anupam Gupta, Aditya Krishnan, and Alireza Samadian for extensive helpful discussions.

References

1. Calgary ParkPlus Homepage, https://www.calgaryparking.com/parkplus
2. SFpark Homepage, http://sfpark.org/
3. SFpark Wikipedia page, https://en.wikipedia.org/wiki/SFpark

4. Antoniadis, A., Barcelo, N., Nugent, M., Pruhs, K., Scquizzato, M.: A $o(n)$-competitive deterministic algorithm for online matching on a line. In: Workshop on Approximation and Online Algorithms. pp. 11–22 (2014)
5. Antoniadis, A., Fischer, C., Tönnis, A.: A collection of lower bounds for online matching on the line. In: Latin American Symposium on Theoretical Informatics. LNCS, vol. 10807, pp. 52–65. Springer (2018)
6. Arora, S., Hazan, E., Kale, S.: The multiplicative weights update method: A meta-algorithm and applications. Theory oF Computing **8**, 121–164 (2012)
7. Arora, S., Hazan, E., Kale, S.: The multiplicative weights update method: a meta-algorithm and applications. Theory of Computing **8**(6), 121–164 (2012). https://doi.org/10.4086/toc.2012.v008a006, http://www.theoryofcomputing.org/articles/v008a006
8. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O.: On-line routing of virtual circuits with applications to load balancing and machine scheduling. Journal of the ACM **44**(3) (May 1997)
9. Azar, Y., Kalyanasundaram, B., Plotkin, S.A., Pruhs, K., Waarts, O.: On-line load balancing of temporary tasks. Journal of Algorithms **22**(1), 93–110 (1997)
10. Bansal, N., Buchbinder, N., Gupta, A., Naor, J.: A randomized $O(\log^2 k)$-competitive algorithm for metric bipartite matching. Algorithmica **68**(2), 390–403 (2014)
11. Bartal, Y.: Probabilistic approximation of metric spaces and its algorithmic applications. In: Symposium on Foundations of Computer Science. pp. 184–193 (1996)
12. Bartal, Y.: On approximating arbitrary metrics by tree metrics. In: ACM Symposium on Theory of Computing. pp. 161–168 (1998)
13. Bender, M., Gilbert, J., Krishnan, A., Pruhs, K.: Competitively pricing parking in a tree (WINE). Lecture Notes in Computer Science, vol. 12495, pp. 220–233. Springer (2020)
14. Cohen, I.R., Eden, A., Fiat, A., Jez, L.: Pricing online decisions: Beyond auctions. CoRR **abs/1504.01093** (2015), http://arxiv.org/abs/1504.01093
15. Cohen, I.R., Eden, A., Fiat, A., Jez, L.: Pricing online decisions: Beyond auctions. In: ACM-SIAM Symposium on Discrete Algorithms. pp. 73–91 (2015)
16. Cohen, I.R., Eden, A., Fiat, A., Jez, L.: Dynamic pricing of servers on trees. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. LIPIcs, vol. 145, pp. 10:1–10:22 (2019)
17. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. Journal of Computer and System Sciences **69**(3), 485–497 (2004)
18. Feldman, M., Fiat, A., Roytman, A.: Makespan minimization via posted prices. In: ACM Conference on Economics and Computation. pp. 405–422 (2017)
19. Fuchs, B., Hochstättler, W., Kern, W.: Online matching on a line. Theoretical Computer Science **332**(1-3), 251–264 (2005)
20. Gupta, A., Lewi, K.: The online metric matching problem for doubling metrics. In: International Colloquium on Automata, Languages, and Programming. pp. 424–435 (2012)
21. Im, S., Moseley, B., Pruhs, K., Stein, C.: Minimizing maximum flow time on related machines via dynamic posted pricing. In: European Symposium on Algorithms. pp. 51:1–51:10 (2017)
22. Kalyanasundaram, B., Pruhs, K.: Online weighted matching. Journal of Algorithms **14**(3), 478–488 (1993)
23. Khuller, S., Mitchell, S.G., Vazirani, V.V.: On-line algorithms for weighted bipartite matching and stable marriages. Theoretical Computer Science **127**(2), 255–267 (1994)
24. Koutsoupias, E., Nanavati, A.: The online matching problem on a line. In: Workshop on Approximation and Online Algorithms. Lecture Notes in Computer Science, vol. 2909, pp. 179–191. Springer (2003)
25. Meyerson, A., Nanavati, A., Poplawski, L.J.: Randomized online algorithms for minimum metric bipartite matching. In: ACM-SIAM Symposium on Discrete Algorithms. pp. 954–959 (2006)
26. Nayyar, K., Raghvendra, S.: An input sensitive online algorithm for the metric bipartite matching problem. In: Symposium on Foundations of Computer Science. pp. 505–515 (2017)
27. Peserico, E., Scquizzato, M.: Matching on the line admits no $o(\sqrt{\log n})$-competitive algorithm. CoRR **abs/2012.15593** (2020), https://arxiv.org/abs/2012.15593
28. Raghvendra, S.: Optimal analysis of an online algorithm for the bipartite matching problem on a line. In: Symposium on Computational Geometry. LIPIcs, vol. 99, pp. 67:1–67:14 (2018)
29. Shoup, D., Pierce, G.: SFpark: Pricing Parking by Demand (2013), https://www.accessmagazine.org/fall-2013/sfpark-pricing-parking-demand/