Min-cost Bipartite Perfect Matching with Delays*

Itai Ashlagi† Yossi Azar‡ Moses Charikar§ Ashish Chiplunkar¶ Ofir Geri‖
Haim Kaplan** Rahul Makhijani†† Yuyi Wang‡‡ Roger Wattenhofer§§

Abstract

In the min-cost bipartite perfect matching with delays (MBPMD) problem, requests arrive online at n points of a finite metric space. Each request is either positive or negative and has to be matched to a request of opposite polarity. As opposed to traditional online matching problems, the algorithm does not have to serve requests as they arrive, and may choose to match them later at a cost. Our objective is to minimize the sum of the distances between matched pairs of requests (the connection cost) and the sum of the waiting times of the requests (the delay cost). This objective exhibits a natural tradeoff between minimizing the distances and the cost of waiting for better matches. This tradeoff appears in many real-life scenarios, notably, ride-sharing platforms. MBPMD is related to its non-bipartite variant, min-cost perfect matching with delays (MPMD), in which each request can matched to any other request. This problem was introduced by Emek et al. (STOC'16), who showed an $O(\log^2 n + \log \Delta)$-competitive randomized algorithm on n-point metric spaces with aspect ratio Δ, using random embedding into tree metrics.

Our contribution is threefold. First, we adapt the algorithm of Emek et al. to the bipartite case, and provide a simplified analysis that improves the competitive ratio to $O(\log n)$. The key ingredient of the algorithm is an $O(h)$-competitive randomized algorithm for MBPMD on weighted trees of height h. Second, we provide an $O(h)$-competitive deterministic algorithm for MBPMD on weighted trees of height h. Third, we show a lower bound of $\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ on the competitive ratio of any randomized algorithm for MBPMD. Along the way, we show that the same construction also improves the lower bound on MPMD from $\Omega(\sqrt{\log n})$ (Azar et al., SODA'17) to $\Omega\left(\frac{\log n}{\log\log n}\right)$.

*February 2016. Some of the results of this paper appear in the arXiv paper [5] (that arXiv paper also contains the results of [6]).

†Management Science & Engineering Department, Stanford University; iashlagi@stanford.edu

‡School of Computer Science, Tel Aviv University; azar@tau.ac.il

§Computer Science Department, Stanford University; moses@cs.stanford.edu

¶School of Computer Science, Tel Aviv University; ashish.chiplunkar@gmail.com

‖Computer Science Department, Stanford University; ofirgeri@cs.stanford.edu

**School of Computer Science, Tel Aviv University; haimk@post.tau.ac.il

††Management Science & Engineering Department, Stanford University; rahulmj@stanford.edu

‡‡Department of Information Technology and Electrical Engineering, ETH Z'urich; yuwang@ethz.ch

§§Department of Information Technology and Electrical Engineering, ETH Z'urich; wattenhofer@ethz.ch

1 Introduction

In the era of mobile Internet access, countless matching problems are solved every day, in a variety of applications, ranging from matching opponents in online games to ride-sharing and online dating. A fundamental problem that arises in these matching platforms is how much delay should an agent incur in order to improve the quality of their match. Consider the case of matching passengers to drivers in ride-sharing platforms such as Uber and Lyft. When a passenger is looking for a ride, they can be matched with any one of many available drivers, who can each be located at a different distance from the passenger. A greedy solution to the matching problem is to match each passenger right when they arrive to the closest available driver. However, this can lead to a very inefficient assignment, if for example, right after the passenger is matched, a new driver becomes available in the same location. Keeping in mind that waiting is also a cost incurred by the participants, we see that this problem exhibits an inherent tradeoff between minimizing the distances and the cost of waiting for better matches.

In this work, we explore this tradeoff and find online algorithms to match requests of two types while minimizing the sum of the distances and the delay costs. The fact that we could make the passengers wait for better matches (and incur a cost for that) is a key property of the above example. This property differentiates our problem from traditional online matching problems, where the main constraint is that vertices must be matched immediately when they arrive. Our problem, on the contrary, falls into the realm of online algorithms with delayed service — a notion introduced in a recent paper by Emek et al. [13] — in which requests need not be served immediately, but can wait (at a cost) in order to get a more efficient solution. The idea of delayed service is not limited to matching problems, and can lead to a better model of real-life scenarios in many problems.

In the problem studied here, requests arrive in an online manner at n points in a finite metric space. Each request is identified by its time of arrival, its location, and its polarity, which can be either positive or negative. Each request can only be matched to a request of opposite polarity. In our motivating example, the requests correspond to the drivers and passengers, who arrive at different times and locations. The polarities of the requests enforce the limitation that passengers only want to be matched to drivers, and vice versa. The objective is to minimize the sum of the *delay cost*, which is the time since the arrival of each request until it is matched, and the *connection cost*, which is the sum of distances between each two requests that are matched to each other. Note that multiple requests may arrive at the same location.

We call this problem *min-cost bipartite perfect matching with delays* (MBPMD), as the requests can be represented as a bipartite graph with edge weights that correspond to the distances in the metric space. We measure the performance of any matching algorithm using the notion of competitive ratio: an algorithm is α-competitive if for every input, the cost incurred by the algorithm is at most α times the cost of the optimal solution. MBPMD is an extension of the problem studied by Emek et al. [13], min-cost perfect matching with delays (MPMD), in which all the requests are of the same type and each can be matched to any other request. They provided a randomized algorithm with competitive ratio of $O(\log^2 n + \log \Delta)$ using probabilistic embedding into tree metrics (Δ denotes the aspect ratio of the metric space). MPMD was studied further by Azar et al. [6], who recently found an algorithm that improved the competitive ratio to $O(\log n)$ and showed a lower bound of $\Omega\left(\sqrt{\log n}\right)$.

The bipartite version of MPMD is more natural in many applications: matching in ride-sharing platforms, job markets, and any situation where there are two types of entities that need to be matched to each other. While MPMD and MBPMD seem quite similar, there is no clear reduction from MBPMD to MPMD, and the study of MBPMD results in a more technically involved analysis. For example, an issue that arises only in MBPMD is that there can be many requests with the

same polarity waiting at the same point without being able to match. In contrast, any reasonable algorithm for MPMD will have at most one request waiting at each point, as it would immediately match requests that are waiting at the same location.

Our Contribution. Our contribution has three parts. First, we adapt the randomized algorithm of Emek et al. [13] to MBPMD and provide a simplified analysis that results in a competitive ratio of $O(\log n)$. This type of simplified analysis can be applied to the non-bipartite case as well. The randomized algorithm for MBPMD consists of a preprocessing phase, in which the finite metric space is embedded into a tree metric, and of a randomized greedy algorithm that solves MBPMD on tree metrics. Informally, we say that an algorithm A is (β, γ)-competitive if for every benchmark algorithm A^*, the (expected) cost incurred by A is at most β times the connection cost of A^* plus γ times the delay cost of A^*. Our analysis shows that the randomized greedy algorithm is $(3, 6h + 1)$-competitive, where h is the height of the tree.

Using probabilistic embedding into HSTs [9, 10, 15] and the height reduction step of Bansal et al. [8], we can embed any finite metric space into a tree metric with height $O(\log n)$ and expected distortion $O(\log n)$. Using this embedding, we can turn any $(O(1), O(h))$-competitive algorithm for tree metrics into a $O(\log n)$-competitive algorithm for any finite metric space.

Our second contribution is a *deterministic* $(10, 10h)$-competitive algorithm for MBPMD on tree metrics. We remark that this algorithm gives rise to a *barely random* algorithm for MBPMD on general metrics, that is, the number of random bits it uses is independent of the input size.

The third part of our contribution is a lower bound on the competitive ratio of randomized algorithms for MBPMD. We show that in a metric space containing n equally spaced points in the unit interval, the competitive ratio for any randomized algorithm for MBPMD is at least $\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$. As a bonus, our construction provides a lower bound of $\Omega\left(\frac{\log n}{\log\log n}\right)$ on the competitive ratio of any randomized algorithm for MPMD, which improves the current lower bound of $\Omega(\sqrt{\log n})$ shown by Azar et al. [6], and matches their upper bound up to the $\log\log n$ factor.

Related Work. The most relevant papers in the context of this work are [6, 13], who studied min-cost bipartite matching with delays (MPMD) in the non-bipartite case. As mentioned above, Emek et al. [13] introduced the notion of online problems with delayed service and provided an $O\left(\log^2 n + \log \Delta\right)$-competitive algorithm for MPMD. Azar et al. [6] provided an $O(h)$-competitive deterministic algorithm for MPMD on tree metrics, and used it to show an $O(\log n)$-competitive algorithm for general metrics. Additionally, they provided a lower bound of $\Omega\left(\sqrt{\log n}\right)$ on the competitive ratio of randomized algorithms for MPMD.

Recent papers in the economics and operations literatures studied matching with delays in stochastic and more structural environments. Anderson et al. [3] and Ashlagi et al. [4] study a model with an underlying stochastic graph and assume agents arrive according to some process. They seek to minimize agents' average waiting time and find that the greedy matching is asymptotically optimal. Akbarpour et al. [2] allow for agents departures and find that when the departure times are known, greedy matching leads to a suboptimal match rate. All these papers do not have the notion of distance. Baccara et al. [7] look at a two-sided market where on each side agents can be one of two types and one type is of higher "quality" than the other. They assume a single agent on each side arrives every time period and find that the optimal matching policy cumulates agents up to a certain threshold.

Online bipartite matching, in general, is an extremely popular model. In the original problem studied by Karp et al. [18], vertices on one side of a bipartite arrive are known in advance and

vertices on the other side arrive online. Each vertex on the online side can match to only some of the offline vertices, and has to matched *upon arrival*. The goal is to maximize the number of matched vertices. There are many extensions and variants of this problem: maximum vertex-weighted matching [1, 12], the AdWords problem [21], metric minimum capacitated assignment [17], and others. The literature on online matching is extensive; see [20] for a survey.

A similar problem is online metric matching, in which requests that arrive online have to be matched to an available server upon their arrival (the locations of the servers are known in advance). Both the requests and the servers belong to a metric space, and the goal is to minimize the sum of the distances between each request and the server it is matched to. This problem is studied in [16] and [19]. Note that MBPMD differs from online metric matching in three ways: the servers also arrive online, the matching of a request can be delayed after its arrival, and the delay cost is also part of the objective function.

2 Preliminaries

A *metric space* $\mathcal{M}$ is a set S equipped with a distance function $d : S \times S \longrightarrow \mathbb{R}^{\geq 0}$ such that $d(x, y) = 0$ if and only if $x = y$, $d(x, y) = d(y, x)$ for all $x, y \in S$, and $d(x, y) + d(y, z) \geq d(x, z)$ for all $x, y, z \in S$. The problem of min-cost bipartite perfect matching with delays (MBPMD) is an online problem defined on an underlying finite metric space $\mathcal{M} = (S, d)$ as follows. An online input instance I over S is a sequence of requests $\langle (p_i, b_i, t_i) \rangle_{i=1}^{m}$, where p_i is a point in the metric space, $b_i \in \{+1, -1\}$, and t_i is the time at which the request arrives. We assume that the number of *positive* requests ($b_i = +1$) equals the number of *negative* requests ($b_i = -1$). The algorithm is required to output a perfect matching between the positive requests and the negative requests. For each pair (i, j) of requests output by the algorithm at time t (where $t \geq \max(t_i, t_j)$), the algorithm pays a connection cost of $d(p_i, p_j)$ and a delay cost of $(t - t_i) + (t - t_j)$. The offline connection cost of creating the pair (i, j) is $d(p_i, p_j)$, and the offline delay cost is $|t_i - t_j|$.

3 The $O(\log n)$ Upper Bound: Overview

Our focus in this section is to give an algorithm for MBPMD on arbitrary metrics, and thus, to prove the following result.

Theorem 1. *There exists a randomized online algorithm with a competitive ratio of $O(\log n)$ for MBPMD on n-point metric spaces.*

As stated previously, we establish the above theorem by reducing MBPMD on arbitrary metrics to MBPMD on tree metrics. A tree metric is given by a tree with positive edge weights such that the points of the metric are the vertices of the tree and the distance between two points is the length of the simple path connecting them. To achieve the reduction, we use the following result (Lemma 3.1 of [6]), which is an easy consequence of probabilistic embedding into tree metrics [15] and Lemma 5.1 of [8].

Lemma 1. *Any n-point metric space $\mathcal{M}$ can be embedded, with distortion $O(\log n)$, into a distribution $\mathcal{D}$ supported on metrics induced by trees of height $O(\log n)$.*

Informally, the *distortion* of an embedding is an upper bound on the expected blowup in the distances between pairs of points.

Analogous to Azar et al. [6], we use the more general notion of (β, γ)-*competitiveness* in addition to the usual notion of competitive ratio. Reusing their notation, given an instance I of MBPMD

and an arbitrary solution SOL of I, we let SOL_d denote its connection cost with respect to the metric d, SOL_t denote its delay cost, and (with a slight abuse of notation) SOL denote its total cost. We restate the definition of (β, γ)-competitiveness for the sake of completeness.

Definition 1. Given a randomized online algorithm $\mathcal{A}$ for MBPMD on a metric space $\mathcal{M} = (S, d)$ and an instance I on S, $\mathcal{A}(I)$ denotes the expected cost of $\mathcal{A}$ on I. $\mathcal{A}$ is said to be α-*competitive* if for every I and every solution SOL of I, $\mathcal{A}(I) \leq \alpha \cdot \mathrm{SOL}$. $\mathcal{A}$ is said to be (β, γ)-*competitive* if for every I and every solution SOL of I, $\mathcal{A}(I) \leq \beta \cdot \mathrm{SOL}_d + \gamma \cdot \mathrm{SOL}_t$.

Given an embedding of a metric space into another with distortion μ, and a (β, γ)-competitive algorithm for the embedding metric, it is easy to see that it can be turned into a $(\mu\beta, \mu\gamma)$-competitive algorithm for the original metric. However, Emek et al. [13] observed that this can strengthened slightly to the following lemma, whose proof is deferred to Appendix A.

Lemma 2. *Suppose that a metric space $\mathcal{M} = (S, d)$ can be embedded into a distribution $\mathcal{D}$ supported on metric spaces over a set $S' \supseteq S$ with distortion μ. Additionally, suppose that for every metric space $\mathcal{M}'$ in the support of $\mathcal{D}$, there is a (possibly randomized) online (β, γ)-competitive algorithm $\mathcal{A}^{\mathcal{M}'}$ for MBPMD on $\mathcal{M}'$. Then there is a $(\mu\beta, \gamma)$-competitive (and thus, $(\max(\mu\beta, \gamma))$-competitive) algorithm $\mathcal{A}$ for MBPMD on $\mathcal{M}$.*

In the next two sections, we give two online algorithms for MBPMD, both of which are $(O(1), O(h))$-competitive on metrics given by edge-weighted trees of height h. The proof of Theorem 1 then follows easily.

Proof of Theorem 1. Given an n-point metric space $\mathcal{M}$, using Lemma 1, embed it into a distribution $\mathcal{D}$ over metrics given by edge-weighted trees of height $O(\log n)$ with distortion $O(\log n)$. The algorithms for tree metrics from the next two sections are $(O(1), O(\log n))$-competitive for every tree metric in the support of $\mathcal{D}$ (Theorems 2 and 3). Therefore, by Lemma 2, there is an $O(\log n)$-competitive algorithm for MBPMD on every metric $\mathcal{M}$. □

Notation Before proceeding to the algorithms for MBPMD on tree metrics, we state here some notation that will be used in their description and analysis. Suppose the tree metric is given by an edge-weighted tree T rooted at an arbitrary vertex r. For a vertex u, let T_u denote the maximal subtree of T rooted at u, e_u denote the edge between u and its parent, and d_u denote the weight of e_u (d_r is defined to be infinite). Similarly, for $e = e_u$ we also use T_e to denote T_u (the subtree rooted at the lower endpoint of e). Let h be the height of the tree, that is, the maximum of the number of vertices in the path between r and any leaf. We assume, without loss of generality, that the requests are given only at the leaves of T. (If not, we pretend as if each non-leaf vertex u has a child u' at distance zero, which is a leaf, and the requests are given at u' instead of u.) Let $\mathrm{lca}(u, v)$ denote the lowest common ancestor of vertices u and v in the tree. Given an edge-weighted tree T, rooted at vertex r, and a set of requests on the leaves of T, we define the *surplus* of a vertex v to be the number of positive requests minus the number of negative requests in T_v, and denote it by $\mathrm{sur}(v)$. (Note that $\mathrm{sur}(v)$ can be negative.) While comparing the performance of the algorithm with a candidate solution SOL, we use $\mathrm{sur}^*(v)$ to denote the surplus of v when running SOL. We also use $\mathrm{sur}(e)$ and $\mathrm{sur}^*(e)$ to denote the surplus of an edge e. If $e = e_u$, then $\mathrm{sur}(e) = \mathrm{sur}(u)$ and $\mathrm{sur}^*(e) = \mathrm{sur}^*(u)$.

4 A Randomized Algorithm for MBPMD on Trees

In this section, we adapt the randomized algorithm for MPMD on trees presented by Emek et al. [13] to the bipartite case. We present a simplified analysis which shows that the algorithm is $(3, 6h+1)$-competitive. The original analysis was restricted to binary hierarchically well-separated trees, and together with the embedding step resulted in a competitive ratio of $O\left(\log^2 n + \log \Delta\right)$ for general metrics. By lifting the binary HST restriction and using the embedding method of Lemma 1, our analysis improves the competitive ratio to $O(\log n)$. The algorithm appears here as Algorithm 1.

Algorithm 1 A Randomized Algorithm for MBPMD on Tree Metrics

Greedy matching (computed upon each arrival): While there are two unmatched requests of opposite polarities in the same point, match those requests immediately. For all other requests, compute a *tentative* greedy matching as follows:

- Consider the vertices from the leaves to the root. (Formally, choose any order such that each vertex is considered only after all of its children have already been considered.)
- When considering a vertex v, let P be the set of positive requests in T_v that are not tentatively matched yet, and N be the set of negative requests in T_v that are not tentatively matched yet.
- While P and N are both non-empty, tentatively match a request from P to a request from N and remove the requests from P and N. Break ties arbitrarily.

At each infinitesimally small time step $[t, t+dt)$**:** For each two requests p_1, p_2 that are tentatively matched, match these two requests with probability $\frac{dt}{d(p_1,p_2)}$ where $d(p_1, p_2)$ is the length of the simple path between p_1 and p_2 in the tree. In that case, we say that the match is realized.

Remark. Algorithm 1 is described in terms of infinitesimally small discrete time steps. However, it can be also described continuously in the following way. For each two requests p_1, p_2 that are tentatively matched, that match will be realized after waiting a time period of Z where $Z \sim Exp\left(\frac{1}{d(p_1,p_2)}\right)$.

Theorem 2. *Algorithm 1 for MBPMD on tree metrics is* $(3, 6h+1)$*-competitive, and hence,* $(6h+1)$*-competitive.*

The proof of Theorem 2 has two parts. First, we bound the connection cost of Algorithm 1 in terms of the connection and delay costs of any benchmark algorithm SOL (Lemma 3). Second, we show how to bound the delay cost of the algorithm using the connection cost of the algorithm and the delay cost SOL_t (Lemma 4).

We introduce some notation that we use in the proof. Let $T = (V, E)$ be a tree with weight function $w : E \to R^{>0}$ (defining a tree metric (V, d)). Denote the connection cost of Algorithm 1 on (V, d) by ALG_d, the delay cost by ALG_t, the total cost by ALG, and let SOL be any benchmark solution for MBPMD on the same tree metric. Let $\mathrm{ALG}_d(t_1, t_2)$ denote the connection cost of the algorithm only due to matches that occur in the time interval $[t_1, t_2)$, and $\mathrm{ALG}_t(t_1, t_2)$ denote the delay cost incurred by ALG during that time interval. $\mathrm{SOL}_d(t_1, t_2)$ and $\mathrm{SOL}_t(t_1, t_2)$ are defined similarly.

For each edge $e \in E$, let $P_e(t), N_e(t)$ denote the number of unmatched positive and negative requests (respectively) inside T_e in ALG at time t, and define $P^*_e(t), N^*_e(t)$ similarly for SOL. Using

these definitions, at time t, $\mathrm{sur}(e) = P_e(t) - N_e(t)$ and $\mathrm{sur}^*(e) = P^*_e(t) - N^*_e(t)$.

We remark on a few properties of the algorithm. First, each edge e can be used as part of at most $|\mathrm{sur}(e)|$ matches at time t. It may be used for less than $|\mathrm{sur}(e)|$ matches, e.g., if there are not enough requests that can be matched to those waiting in T_e. Second, all the requests from T_e that are matched through e are of the same polarity (otherwise, they would have been matched at a lower level). With these observations, we are ready to prove the key lemma of the section.

Lemma 3. $\mathbb{E}[\mathrm{ALG}_d] \leq \mathrm{SOL}_d + 2h \cdot \mathrm{SOL}_t$

Proof. For each edge $e \in E$, define the following potential at time t:

$$\Phi_e(t) = w(e)\,|\mathrm{sur}(e) - \mathrm{sur}^*(e)| = w(e)\,|P_e(t) - N_e(t) - (P^*_e(t) - N^*_e(t))|$$

The total potential at time t is defined as $\Phi(t) = \sum_{e \in E} \Phi_e(t)$.

We divide the time into intervals. The first interval starts at time 0. An interval ends and the next interval begins when a new request arrives or when SOL matches two requests. Let $[t_1, t_2)$ be an interval and denote $\Delta\Phi = \Phi(t_2) - \Phi(t_1)$. We wish to prove that

$$\mathbb{E}[\mathrm{ALG}_d(t_1, t_2) + \Delta\Phi] \leq \mathrm{SOL}_d(t_1, t_2) + 2h \cdot \mathrm{SOL}_t(t_1, t_2).$$

There are three events that can happen: the arrival of a request, a match by SOL, or a match by ALG. Arrivals and matches by SOL can only happen at time t_1 during the interval. Matches by ALG can happen at any time in (t_1, t_2) (the probability that a match occurs at time t_1 is 0). We consider each of these events.

Arrival. We claim that the arrival of a request does not change the potential. For each edge e in the path from the request to the root, either P_e and P^*_e or N_e and N^*_e increase by 1, and Φ_e remains the same. If ALG or SOL matches the new request to another request at the same location, the surplus (and also the potential) does not change.

Match by SOL. The connection cost $\mathrm{ALG}_d(t_1, t_2)$ is not affected by the actions of SOL. Note that the connection cost incurred by SOL due to a single match is the sum of weights of edges that are used as part of the match, and that the match only changes the potential of these edges. For each edge e that is used as part of a match, $\Delta\Phi_e \leq w(e)$. By summing over all these edges, we get $\Delta\Phi \leq \mathrm{SOL}_d$.

Match by ALG. At any time $t \in (t_1, t_2)$, there are no arrivals or matches in SOL. Hence, the tentative matching maintained by the algorithm does not change, and the potential can only change due to a match by ALG. If a match of a pair of requests is realized by ALG, the potential of each edge e in the path connecting these two requests either increases or decreases by $w(e)$.

Let e be an edge that is used in the tentative matching, and denote by $\mathrm{ALG}_e(t, t')$ the connection cost that ALG incurred between (t, t') due to edge e. The following claim relates the expected connection cost and change in potential at edge e to the surplus in SOL and to the length of the interval $[t_1, t_2)$, which we will relate to the delay cost of SOL. Note that $|\mathrm{sur}^*(e)|$ does not change during (t_1, t_2) (as there are no arrivals or matches in SOL). The proof of the claim is deferred to Appendix B.

Claim 1. *For every edge e that is used in the tentative matching,* $\mathbb{E}[\mathrm{ALG}_e(t_1, t_2) + \Delta\Phi_e] \leq 2\,|\mathrm{sur}^*(e)|\,(t_2 - t_1)$.

The claim asserts that the expected connection cost due to the use of e and the change in Φ_e is at most $2\,|\mathrm{sur}^*(e)| \cdot (t_2 - t_1)$. Now we claim that there are least $|\mathrm{sur}^*(e)|$ requests waiting in SOL in the subtree of e. This follows from the fact that $\max\{P^*_e(t), N^*_e(t)\} \geq |\mathrm{sur}^*(e)|$. The delay cost $\mathrm{SOL}_t(t_1, t_2)$ due to these requests is $|\mathrm{sur}^*(e)|\,(t_2 - t_1)$.

We have shown that for every edge e, we can "charge" the sum of the expected connection cost incurred by ALG and the change in potential to the requests waiting in SOL in T_e: this sum is at most $2\,|\mathrm{sur}^*(e)|\,(t_2 - t_1)$, while there are at least $|\mathrm{sur}^*(e)|$ requests waiting in SOL each leading to a delay cost of $t_2 - t_1$. A request waiting in SOL is charged at most once for each edge on the path that connects the request to the root of the tree, that is, each request is charged at most h times.

Summing over all the edges, we get that during the interval $[t_1, t_2)$,

$$\mathbb{E}[\mathrm{ALG}_d(t_1, t_2) + \Delta\Phi] \leq \mathrm{SOL}_d(t_1, t_2) + 2h \cdot \mathrm{SOL}_t(t_1, t_2).$$

The lemma follows by summing these inequalities for all intervals and by noticing that the potential is 0 at time 0 and after both algorithms have matched all the requests. □

The following lemma is similar to Lemma 7 in [13] (Lemma 4.8 in the full version [14]). The proof is deferred to Appendix B.

Lemma 4. $\mathbb{E}[\mathrm{ALG}_t] \leq 2\mathbb{E}[\mathrm{ALG}_d] + \mathrm{SOL}_t$

Proof of Theorem 2. From Lemma 4, we get that $\mathbb{E}[\mathrm{ALG}] = \mathbb{E}[\mathrm{ALG}_t] + \mathbb{E}[\mathrm{ALG}_d] \leq 3\mathbb{E}[\mathrm{ALG}_d] + \mathrm{SOL}_t$. From Lemma 3, $\mathbb{E}[\mathrm{ALG}_d] \leq \mathrm{SOL}_d + 2h \cdot \mathrm{SOL}_t$. We conclude that $\mathbb{E}[\mathrm{ALG}] \leq 3\,\mathrm{SOL}_d + (6h + 1)\,\mathrm{SOL}_t$. □

5 A Deterministic Algorithm for MBPMD on Trees

The algorithm, which appears here as Algorithm 2, maintains two forests, F^+ and F^-, both initialized to be empty. For every vertex u, the algorithm also maintains two counters, z_u^+ and z_u^-, initially set to zero. Intuitively, if $e_u \in F^+$ (resp. $e_u \in F^-$), then e_u is available for connecting a positive (resp. negative) request inside T_u to a negative (resp. positive) request outside. We say that a vertex u is *positively saturated* (resp. *negatively saturated*) if the edge e_u is in F^+ (resp. F^-), else, we say it is *positively unsaturated* (resp. *negatively unsaturated*). The root r is always positively as well as negatively unsaturated, by definition. Note F^+ and F^- are not necessarily disjoint, and therefore, a vertex can be both positively as well as negatively saturated at the same time. The rest of the section is dedicated to proving the following theorem that bounds the competitive ratio of Algorithm 2.

Theorem 3. *Algorithm 2 for MBPMD on tree metrics is* $(10, 10h)$*-competitive, and hence,* $10h$*-competitive.*

For any vertex u, we divide time into phases as follows. The first phase at u starts when the algorithm starts. Whenever the edge e_u is used to connect requests, the phase at u ends and a new phase begins at u. Note that the last phase at any u is necessarily incomplete, and that the phases at different vertices need not be aligned. Observe that at the beginning of any phase at u, both z_u^+ and z_u^- are zero, whereas at the end, one of them is equal to $2d_u$ and the other is at most $2d_u$.

For the analysis, imagine a variable y_u^+ (resp. y_u^-) for every u, which increases at the same rate as z_u^+ (resp. z_u^-) during the run of the algorithm, but which is never reset to zero. We will separately relate the connection cost as well as the delay cost of the algorithm to $\sum_u (y_u^+ + y_u^-)$, and then relate $\sum_u (y_u^+ + y_u^-)$ to the cost of an arbitrary solution SOL, and thus, prove $(O(1), O(h))$-competitiveness.

Lemma 5. *The connection cost of the algorithm is at most* $\frac{1}{2}\sum_u (y_u^+ + y_u^-)$.

Proof. For an arbitrary vertex u, recall that every usage of edge e_u, which results in a connection cost of d_u, marks the end of a phase at u. In every phase at u, one of z_u^+ and z_u^- increases from 0 to $2d_u$. Thus, in every phase at u, $y_u^+ + y_u^-$ increases by at least $2d_u$. This implies the claim. □

Algorithm 2 A Deterministic Algorithm for MBPMD on Tree Metrics

Initialize: $F^+ := \emptyset$, $F^- := \emptyset$. For each vertex u, $z_u^+ := 0$ and $z_u^- := 0$.
At every moment:

- While there are two unmatched requests of opposite polarities at the same point, match those requests immediately.
- For each vertex u, if u is positively unsaturated and $\mathrm{sur}(u) > 0$ (resp. u is negatively unsaturated and $\mathrm{sur}(u) < 0$), then increase counter z_u^+ (resp. z_u^-) at the rate $\mathrm{sur}(u)$ (resp. $-\mathrm{sur}(u)$). Else, keep the counter frozen.
- For each vertex $u \neq r$, as soon as the value of z_u^+ (resp. z_u^-) becomes equal to $2d_u$, add the edge e_u to F^+ (resp. F^-). This makes u positively saturated (resp. negatively saturated), and z_u^+ (resp. z_u^-) is frozen.
- For each positive request, located at u, and each negative request, located at v, as soon as the entire path between u and $\mathrm{lca}(u, v)$ is contained in F^+, and the entire path between v and $\mathrm{lca}(u, v)$ is contained in F^-,
  - Connect the request at u to the request at v.
  - Remove the edges on the path from u to v from both F^+ as well as F^-.
  - For every vertex $w \neq \mathrm{lca}(u, v)$ on the path from u to v, reset $z_w^+ := 0$ and $z_w^- := 0$. (All these vertices are unsaturated due to the previous step.)

Lemma 6. *The delay cost of the algorithm is at most* $2\sum_u (y_u^+ + y_u^-)$.

We defer the proof of the above lemma to Appendix C. Now we need to relate the value $\sum_u (y_u^+ + y_u^-)$ at the end of the algorithm's run to the cost of an arbitrary solution SOL to the instance. For this, let x_u be the total delay cost incurred by SOL due to requests inside T_u, and x'_u be the total connection cost incurred by SOL for using the edge e_u.

Lemma 7. *At the end of the algorithm's run, for all vertices* u, $y_u^+ + y_u^- \leq 4(x_u + x'_u)$.

We defer the proof to Appendix C. Next, we relate $\sum_u (x_u + x'_u)$ to the cost of the solution SOL. Denoting the distance function of the tree metric by d, recall that SOL_d and SOL_t denote the connection cost and the delay cost of SOL, respectively. Our final ingredient is Lemma 3.6 from [6], stated as follows.

Lemma 8. $\sum_u (x_u + x'_u) \leq \mathrm{SOL}_d + h \cdot \mathrm{SOL}_t$.

The competitiveness of the algorithm now follows easily.

Proof of Theorem 3. From Lemmas 5 and 6, the algorithm's total cost is at most $\frac{5}{2}\sum_u (y_u^+ + y_u^-)$. By Lemma 7, this is at most $10\sum_u (x_u + x'_u)$, which by Lemma 8, is at most $10\,\mathrm{SOL}_d + 10h \cdot \mathrm{SOL}_t$. Therefore, the algorithm is $(10, 10h)$-competitive. □

6 The Lower Bounds

The focus of this section is to prove the following lower bound results.

Theorem 4. *There is an n-point metric space on which any randomized algorithm for MBPMD has competitive ratio $\Omega(\sqrt{\log n / \log\log n})$ against an oblivious adversary.*

Theorem 5. *There is an n-point metric space on which any randomized algorithm for MPMD has competitive ratio $\Omega(\log n / \log\log n)$ against an oblivious adversary.*

To prove a lower bound of α on the competitive ratio of randomized online algorithms, we use Yao's min-max principle [11, 22, 23], and give a distribution over input instances which defeats every deterministic algorithm by the factor α. The required distributions for proving the above two lower bounds are very similar; a random MBPMD instance is generated by generating a random MPMD instance and giving polarities to the requests in a randomized fashion. Due to the inherent similarity, we merge the descriptions of the two distributions.

The metric space is given by a parameter L, which is an even integer. Let $n = 2L^{\left\lfloor \frac{L}{\log_2 L} \right\rfloor} \leq 2^{L+1}$, so that $L = \Theta(\log n)$. The required metric space consists of n equally spaced points on the real interval $[0, 1]$. All asymptotic notation in this section is with respect to $n \to \infty$, or equivalently $L \to \infty$.

Every instance in the support of the distribution consists of requests given in $r = \lfloor L / \log_2 L \rfloor$ phases. In each phase, the requests are given at once at the beginning, and they are equally spaced in $[0, 1]$. Furthermore, the set of points at which these requests are given is a suitably chosen random subset of the set of points at which requests were given in the previous phase. The number of requests in phase i is $n_i = 2L^{r-i}$, and the duration of phase i is $t_i = 1/L^{r-i}$ time units. The distribution $\mathcal{D}$ on M(B)PMD instances is generated as follows.

- Let $r := \lfloor L / \log_2 L \rfloor$, $n := 2L^r$, $S_0 := \{1/n, 2/n, \ldots, 1\}$.
- For $i = 0, \ldots, r$,
  1. **Only for MBPMD:** Choose b_i uniformly at random from $\{+1, -1\}$.
  2. Give requests at points in S_i. **Only for MBPMD:** Starting with the polarity b_i for the leftmost request in S_i, assign alternating polarities to the requests.
  3. Index the points in S_i from left to right, with index 1 for the leftmost point. Construct sets $Y_0^{i+1}, Y_1^{i+1} \subseteq S_i$ as follows.
     (a) Y_0^{i+1} is the set of points whose index is an integer multiple of L.
     (b) Y_1^{i+1} is the set of points whose index is an integer multiple of L plus $L/2$, that is, $L/2$, $3L/2$, and so on. (Recall that L is even.)
  4. Choose z_{i+1} uniformly at random from $\{0, 1\}$. Let $S_{i+1} := Y_{z_{i+1}}^{i+1}$. (Thus, $|S_{i+1}| = |Y_0^{i+1}| = |Y_1^{i+1}| = |S_i|/L$.)
  5. Wait for time $t_i = 1/L^{r-i}$ (and then move on to the next phase, if $i < r$).

In order to bound the expected cost of an arbitrary deterministic M(B)PMD algorithm, we need to set up some notation and prove a key lemma. For a set S of requests on an underlying metric space, and a real number c, let $\mathrm{MIN}(S, c)$ denote the cost of the min-cost (possibly partial) matching on S (ignoring signs, even in MBPMD), where the cost of a matching is the sum of distances between the matched pairs of points, plus a penalty of c per unmatched request. The following lemma can be thought of as a triangle inequality on sets of requests. We defer its proof to Appendix D.

Lemma 9. *Let X, Y_0, and Y_1 be arbitrary sets of requests on an underlying metric space. Then $\mathrm{MIN}(X \cup Y_0, c) + \mathrm{MIN}(X \cup Y_1, c) \geq \mathrm{MIN}(Y_0 \cup Y_1, c)$.*

We use the above lemma to prove the following lower bound on the cost of an arbitrary deterministic online M(B)PMD algorithm in every phase.

Lemma 10. *Every deterministic online M(B)PMD algorithm incurs a cost of at least* $1/4$ *in expectation in every phase* i*, conditioned on* $z_1, \ldots, z_{i-1}$ *(and* $b_0, \ldots, b_{i-1}$*, additionally, for MBPMD).*

Proof. Let X be the set of pending requests from earlier stages at the beginning of an arbitrary phase i. If we condition on the random events from the previous phases, X is fixed. Recall that t_i, the duration of the phase, is $1/L^{r-i}$. Since no new requests arrive while the phase is in progress, we may assume that each request which is matched during the phase is matched at the beginning itself. Thus, each unmatched request waits from the beginning till the end of the phase, resulting in a delay cost of t_i. Hence, the expected cost of the algorithm is at least $\mathbb{E}_{z_i}[\mathrm{MIN}(X \cup S_i, t_i)] = (\mathrm{MIN}(X \cup Y_0^i, t_i) + \mathrm{MIN}(X \cup Y_1^i, t_i))/2$. (This holds even in the case of MBPMD, because $\mathrm{MIN}(X \cup Y_j^i, t_i)$ is the cost of the best possible matching ignoring polarities, whereas the algorithm produces a matching which respects polarities, which can only have a larger cost.) Thus, by Lemma 9, the algorithm's expected cost is bounded from below by $\mathrm{MIN}(Y_0^i \cup Y_1^i, t_i)/2$.

Observe that $Y_0^i \cup Y_1^i$ is a set of $4L^{r-i}$ equi-spaced requests with spacing $1/4L^{r-i}$. Thus, $\mathrm{MIN}(Y_0^i \cup Y_1^i, t_i) = \mathrm{MIN}(Y_0^i \cup Y_1^i, 1/L^{r-i}) = 1/2$, since it is cheaper to match all requests in $Y_0^i \cup Y_1^i$ and pay $1/8L^{r-i}$ per request, rather than paying $1/L^{r-i}$ per unmatched request. Therefore, the cost of the algorithm is at least $\mathrm{MIN}(Y_0^i \cup Y_1^i, t_i)/2 \geq 1/4$ in every phase i. □

Corollary 1. *Every deterministic online M(B)PMD algorithm incurs a cost of at least* $r/4 = \Omega(L/\log L)$ *in expectation on a random input drawn from* $\mathcal{D}$*.*

Proof. Follows by unconditioning the bound from Lemma 10. □

We use two different techniques to prove upper bounds on the cost of the optimum solution for MPMD and MBPMD, resulting in different upper bounds and thus, forcing different lower bounds on the competitive ratio. The upper bounds are given by the following two lemmas, whose proofs are deferred to Appendix D.

Lemma 11. *The expected cost of the optimal solution of an MBPMD instance generated from the distribution* $\mathcal{D}$ *is* $O(\sqrt{L/\log L})$*.*

Lemma 12. *Every MPMD instance generated from* $\mathcal{D}$ *has a solution of cost at most* $1 + 2/\log_2 L = O(1)$*.*

In the case of MBPMD, it turns out that the connection cost of a suitably constructed solution of an instance from $\mathcal{D}$ is related to the outcome of an r-step random walk on the integers, which starts from 0 and takes each step in either direction with equal probability. This results in an expected connection cost of $O(\sqrt{r})$, whereas the delay cost is bounded by the sum of the arrival times of the requests, and turns out to be much smaller. In case of MPMD, a much simpler argument proves that the connection cost is $O(1)$ and the delay cost is $O(1/\log_2 L)$. Using the fact that $L = \Theta(\log n)$, Theorems 4 and 5 follow easily.

Proof of Theorem 4. Follows from Corollary 1 and Lemma 11. □

Proof of Theorem 5. Follows from Corollary 1 and Lemma 12. □

7 Concluding Remarks and Open Problems

In this paper, we showed a randomized $O(\log n)$-competitive algorithm and a lower bound of $\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ on the competitive ratio of any randomized algorithm for MBPMD. One natural open problem is closing the gap between these bounds. Another open question is whether randomization is needed in solving MBPMD. While for trees we provided a deterministic $O(h)$-competitive algorithm, the question of finding deterministic algorithms or lower bounds for general metrics remains open.

We took here the centralized planner's view that can dictate who can match to whom. An interesting open question is what is the efficiency loss if the market is decentralized and agents selfishly decide whether to match with a partner or to wait for a closer partner. There are some modeling decisions to make here, but in general the competitive ratio should increase, since agents will impose negative externalities on others (such analysis is done under stochastic assumptions in [7]).

Our model only scratches the surface of the numerous variants of MBPMD that can be practical for many applications. Keeping the ride-sharing motivating example in mind, one can model carpooling as a many-to-one matching problem while taking into account the different destinations of the passengers. One can also allow requests to move to other points while waiting, simulating drivers that head toward busy areas while waiting for a match.

Further study of MBPMD can involve different assumptions on the input. While we analyzed the competitive ratio for worst-case input, a more refined analysis can be made in the case where the input is drawn from some known distribution. Another possible analysis beyond the worst case is to consider a more restricted family of metric spaces with a structure that may result in better bounds even for worst-case input.

Finally, the delay of services and allocations shows up in many applications, which can be studied using the notion of online problems with delayed service.

Acknowledgment

The authors thank Amos Fiat for his insightful involvement in the discussions.

References

[1] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1253–1264, 2011.

[2] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Thickness and information in dynamic matching markets. *Available at SSRN 2394319*, 2016.

[3] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. A dynamic model of barter exchange. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1925–1933, 2015.

[4] Itai Ashlagi, Maximillien Burq, Patrick Jaillet, and Vahideh Manshadi. On matching and thickness in heterogeneous dynamic markets. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, pages 765–765. ACM, 2016.

[5] Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost (bipartite) perfect matching with delays, 2016. arXiv:1610.05155 [cs.DS].

[6] Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. In *Proceedings of the Twenty-eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1051–1061, 2017.

[7] Mariagiovanna Baccara, SangMok Lee, and Leeat Yariv. Optimal dynamic matching. *Available at SSRN 2641670*, 2015.

[8] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5):40, 2015.

[9] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[10] Yair Bartal. Graph decomposition lemmas and their role in metric embedding methods. In *Algorithms - ESA 2004, 12th Annual European Symposium*, pages 89–97, 2004.

[11] Allan Borodin and Ran El-Yaniv. On randomization in on-line computation. *Inf. Comput.*, 150(2):244–267, 1999.

[12] Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 101–107, 2013.

[13] Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: haste makes waste! In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 333–344, 2016.

[14] Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: Haste makes waste!, 2016. arXiv:1603.03024 [cs.DS].

[15] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[16] Anupam Gupta and Kevin Lewi. The online metric matching problem for doubling metrics. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'12, pages 424–435, 2012.

[17] Bala Kalyanasundaram and Kirk R. Pruhs. The online transportation problem. *SIAM J. Discret. Math.*, 13(3):370–383, May 2000.

[18] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 352–358, 1990.

[19] Elias Koutsoupias and Akash Nanavati. The online matching problem on a line. In *Approximation and Online Algorithms, First International Workshop*, pages 179–191, 2003.

[20] Aranyak Mehta. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.*, 8(4):265–368, October 2013.

[21] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5), 2007.

[22] Leen Stougie and Arjen P. A. Vestjens. Randomized algorithms for on-line scheduling problems: how low can't you go? *Oper. Res. Lett.*, 30(2):89–96, 2002.

[23] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.

A Proof Omitted from Section 3

Recall the following definition of a metric embedding and its distortion.

Definition 2. Let $\mathcal{M} = (S, d)$ be a finite metric space, and let $\mathcal{D}$ be a probability distribution over metrics on a finite set $S' \supseteq S$. We say that $\mathcal{M}$ *embeds* into $\mathcal{D}$ with *distortion* μ if

- For every $x, y \in S$ and every metric space $\mathcal{M}' = (S', d')$ in the support of $\mathcal{D}$, we have $d(x, y) \leq d'(x, y)$.
- For every $x, y \in S$, we have $\mathbb{E}_{\mathcal{M}'=(S',d')\sim\mathcal{D}}[d'(x, y)] \leq \mu \cdot d(x, y)$.

Proof of Lemma 2. Algorithm $\mathcal{A}$ simply samples a metric space $\mathcal{M}' = (S', d')$ from the distribution $\mathcal{D}$, and simulates the behavior of $\mathcal{A}^{\mathcal{M}'}$. Clearly, the delay cost paid by $\mathcal{A}$ is the same as the delay cost paid by $\mathcal{A}^{\mathcal{M}'}$. Furthermore, since $d(p, q) \leq d'(p, q)$ for all $p, q \in S$, the connection cost paid by $\mathcal{A}$ is no more than the connection cost paid by $\mathcal{A}^{\mathcal{M}'}$. Fix an input instance I of MBPMD on $\mathcal{M}$, and an arbitrary solution SOL of I. Then the expected cost $\mathcal{A}(I)$ of the algorithm $\mathcal{A}$ on I is bounded as follows.

$$\mathcal{A}(I) \leq \mathbb{E}_{\mathcal{M}'=(S',d')\sim\mathcal{D}}[\mathcal{A}^{\mathcal{M}'}(I)]$$

where $\mathcal{A}^{\mathcal{M}'}(I)$ is the expected cost of $\mathcal{A}^{\mathcal{M}'}$ on I, the expectation being taken over the randomness internal to $\mathcal{A}^{\mathcal{M}'}$. Since $\mathcal{A}^{\mathcal{M}'}$ is (β, γ)-competitive, we have by definition,

$$\mathcal{A}^{\mathcal{M}'}(I) \leq \beta\,\mathrm{SOL}_{d'} + \gamma\,\mathrm{SOL}_t$$

This implies

$$\begin{aligned}
\mathcal{A}(I) &\leq \mathbb{E}_{\mathcal{M}'=(S',d')\sim\mathcal{D}}[\beta\,\mathrm{SOL}_{d'} + \gamma\,\mathrm{SOL}_t] \\
&= \beta \cdot \mathbb{E}_{\mathcal{M}'=(S',d')\sim\mathcal{D}}[\mathrm{SOL}_{d'}] + \gamma\,\mathrm{SOL}_t \\
&\leq \beta\mu \cdot \mathrm{SOL}_d + \gamma \cdot \mathrm{SOL}_t
\end{aligned}$$

where the equality follows from linearity of expectation and the fact that SOL_t is independent of $\mathcal{M}'$, while the second inequality follows from the definition of distortion. By the definition of (β, γ)-competitiveness, the claim follows. □

B Proofs Omitted from Section 4

Proof of Claim 1. Consider an edge e that is used for (tentatively) matching k positive requests $p_1, \ldots, p_k$ to k negative requests $n_1, \ldots, n_k$. Remember that all the requests in T_e that are matched through e are of the same type. Without loss of generality, assume that the requests in T_e are the positive requests $p_1, \ldots, p_k$, that is, for every match that is realized, P_e decreases by 1. There are two cases.

Case 1: $\mathrm{sur}^*(e) \leq 0$. In that case, since $\mathrm{sur}(e)$ decreases by 1 for each match that is realized, the potential Φ_e decreases by $w(e)$ for each match that ALG makes. Thus, for each match the connection cost incurred by ALG for using edge e and the change in the potential sum to 0. Namely, $\mathbb{E}[\mathrm{ALG}_e(t_1, t_2) + \Delta\Phi_e] = 0 \leq 2\,|\mathrm{sur}^*(e)|\,(t_2 - t_1)$.

Case 2: $\mathrm{sur}^*(e) > 0$. In that case, as long as $\mathrm{sur}(e) > \mathrm{sur}^*(e)$, each match will decrease Φ_e by $w(e)$, and when $\mathrm{sur}(e) \leq \mathrm{sur}^*(e)$, each match will increase Φ_e by $w(e)$. Note that $\mathrm{sur}(e) \geq k > 0$. Let $t' \in [t_1, t_2)$ be the minimal time such that $\mathrm{sur}(e) \leq \mathrm{sur}^*(e)$. If there is no such t', we set $t' = t_2$.

For $1 \leq i \leq k$, let X_i be an indicator random variable for the event that the request p_i is matched during the interval (t', t_2) (conditioned on p_i not being matched before t'), and Z_i be an exponential random variable with parameter $\frac{1}{d(p_i,n_i)}$. Then,

$$\mathbb{E}[X_i] = \Pr[Z_i < t_2 | Z_i > t] = \Pr[Z_i < t_2 - t'] = 1 - e^{-\frac{t_2-t'}{d(p_i,n_i)}} \leq 1 - e^{-\frac{t_2-t_1}{w(e)}} \leq \frac{t_2 - t_1}{w(e)}.$$

Now, note that

$$\begin{aligned}
\mathbb{E}[\mathrm{ALG}_e(t_1, t_2) + \Delta\Phi_e] &= \mathbb{E}[\mathrm{ALG}_e(t_1, t') + \Phi_e(t') - \Phi_e(t_1)] \\
&\quad + \mathbb{E}[\mathrm{ALG}_e(t', t_2) + \Phi_e(t_2) - \Phi_e(t')] \\
&= \mathbb{E}[\mathrm{ALG}_e(t_1, t') + \Phi_e(t') - \Phi_e(t_1)] \\
&\quad + \mathbb{E}[\mathrm{ALG}_e(t', t_2) + \Phi_e(t_2) - \Phi_e(t') | t' < t_2] \Pr[t' < t_2] \\
&\leq \mathbb{E}[\mathrm{ALG}_e(t_1, t') + \Phi_e(t') - \Phi_e(t_1)] \\
&\quad + \mathbb{E}[\mathrm{ALG}_e(t', t_2) + \Phi_e(t_2) - \Phi_e(t') | t' < t_2]
\end{aligned}$$

where we use the facts that $\mathbb{E}[\mathrm{ALG}_e(t', t_2) + \Phi_e(t_2) - \Phi_e(t') | t' = t_2] = 0$ and that $\mathrm{ALG}_e(t', t_2) + \Phi_e(t_2) - \Phi_e(t')$ is non-negative (the potential can only increase due to matches in (t', t_2)).

If ALG makes N_1 matches during (t_1, t'), then $\mathrm{ALG}_e(t_1, t') = N_1 \cdot w(e)$, while $\Phi_e(t') - \Phi_e(t_1) = -N_1 \cdot w(e)$ (before t', the potential only decreases due to the matches). Thus, $\mathbb{E}[\mathrm{ALG}_e(t_1, t') + \Phi_e(t') - \Phi_e(t_1)] = 0$.

Consider $\mathbb{E}[\mathrm{ALG}_e(t', t_2) + \Phi_e(t_2) - \Phi_e(t') | t' < t_2]$. Note that at if $t' < t_2$, then $N_1 = \mathrm{sur}(e) - \mathrm{sur}^*(e)$.[1] Then, during the interval (t', t_2), there are $k - N_1 \leq |\mathrm{sur}^*(e)|$ requests that may be matched using edge e. Denote the number of requests that are matched using e during (t', t_2) by N_2. Intuitively, $\mathbb{E}[N_2]$ is at most $(k - N_1) \cdot \frac{t_2-t_1}{w(e)}$, since for each such request i, $\mathbb{E}[X_i] \leq \frac{t_2-t_1}{w(e)}$.

Formally, if for $S \subseteq \{1, \ldots, k\}$ of size N_1, A_S denotes the event that the requests $\{p_i | i \in S\}$ are matched before t',

$$\begin{aligned}
\mathbb{E}[N_2 | t' < t_2] &= \sum_{\substack{S \subseteq \{1,\ldots,k\}: \\ |S| = N_1}} \mathbb{E}[N_2 | t' < t_2, A_S] \Pr[A_S | t' < t_2] \\
&= \sum_{\substack{S \subseteq \{1,\ldots,k\}: \\ |S| = N_1}} \left(\sum_{i \notin S} \mathbb{E}[X_i | t' < t_2, A_S] \right) \Pr[A_S | t' < t_2] \\
&\leq \sum_{\substack{S \subseteq \{1,\ldots,k\}: \\ |S| = N_1}} \left(\sum_{i \notin S} \frac{t_2 - t_1}{w(e)} \right) \Pr[A_S | t' < t_2] \\
&= (k - N_1) \cdot \frac{t_2 - t_1}{w(e)} \sum_{\substack{S \subseteq \{1,\ldots,k\}: \\ |S| = N_1}} \Pr[A_S | t' < t_2] \\
&= (k - N_1) \cdot \frac{t_2 - t_1}{w(e)} \\
&\leq |\mathrm{sur}^*(e)| \cdot \frac{t_2 - t_1}{w(e)}
\end{aligned}$$

[1] This refers to $\mathrm{sur}(e)$ at time t_1. Note that at most one match is realized at time t' (the probability that two matches will be realized at the same time is 0). Therefore, N_1 must be $\mathrm{sur}(e) - \mathrm{sur}^*(e)$ and not greater than that.

Finally, since for each request matched after t', the potential increases by $w(e)$, we get that

$$\begin{aligned}\mathbb{E}[\mathrm{ALG}_e(t', t_2) + \Phi_e(t_2) - \Phi_e(t')|t' < t_2] &= 2w(e) \cdot \mathbb{E}[N_2|t' < t_2] \\ &\leq |\mathrm{sur}^*(e)| \cdot 2w(e) \cdot \frac{t_2 - t_1}{w(e)} \\ &= 2\,|\mathrm{sur}^*(e)|\,(t_2 - t_1)\end{aligned}$$

and $\mathbb{E}[\mathrm{ALG}_e(t_1, t_2) + \Delta\Phi_e] \leq 2\,|\mathrm{sur}^*(e)|\,(t_2 - t_1)$. □

Proof of Lemma 4. We divide the time into intervals as in the proof of Lemma 3. The first interval starts at time 0. An interval ends and the next interval begins when a new request arrives or when SOL matches two requests. Let $[t_1, t_2)$ be an interval. We show that $\mathbb{E}[\mathrm{ALG}_t(t_1, t_2)] \leq 2\mathbb{E}[\mathrm{ALG}_d(t_1, t_2)] + \mathrm{SOL}_t(t_1, t_2)$.

Note that the number of requests that are not tentatively matched at time t is $|\mathrm{sur}(r)|$, and that at any time t, $\mathrm{sur}(r) = \mathrm{sur}^*(r)$ (both ALG and SOL run on the same input and clear requests in pairs of different types). Intuitively, since SOL also had a surplus of the same number of requests, the delay cost incurred by SOL is at least the delay cost incurred due to the requests that are not tentatively matched by ALG. Formally, note that $\mathrm{sur}(r)$ has the same value at all times $t \in [t_1, t_2)$. Let $K = |\mathrm{sur}(r)|$ for some $t \in [t_1, t_2)$. Then, the delay cost of the requests that are not tentatively matched during $[t_1, t_2)$ is $K \cdot (t_2 - t_1)$ (note that the tentative matching cannot not be recomputed in the middle of an interval). In addition, during $[t_1, t_2)$, SOL has at least K requests waiting, hence $\mathrm{SOL}_t(t_1, t_2) \geq K \cdot (t_2 - t_1)$.

So far we have shown that during $[t_1, t_2)$, the delay cost of the requests that are not tentatively matched is at most $\mathrm{SOL}_t(t_1, t_2)$. We now consider all the requests that are part of the tentative matching computed by ALG. For each pair of requests, we compare the expected connection cost and expected delay cost. Let p_1, p_2 be two requests that are tentatively matched by ALG. We denote the connection cost due to p_1, p_2 during $[t_1, t_2)$ by $\Delta\,\mathrm{ALG}_d(p_1, p_2)$ and the delay cost due to p_1, p_2 during $[t_1, t_2)$ by $\Delta\,\mathrm{ALG}_t(p_1, p_2)$.

The time until p_1, p_2 are matched is an exponential random variable Z with parameter $\frac{1}{d(p_1,p_2)}$. The requests are matched during the interval if $Z < t_2 - t_1$. Then, the expected connection cost is $\mathbb{E}[\Delta\,\mathrm{ALG}_d(p_1, p_2)] = d(p_1, p_2) \cdot \Pr[Z < t_2 - t_1] = d(p_1, p_2)(1 - e^{-\frac{t_2 - t_1}{d(p_1,p_2)}})$. The expected delay cost for each one of p_1, p_2 during $[t_1, t_2)$ is

$$\mathbb{E}[min\{Z, t_2 - t_1\}] = \mathbb{E}[Z - (Z - (t_2 - t_1))^+] = d(p_1, p_2)(1 - e^{-\frac{t_2 - t_1}{d(p_1,p_2)}}).$$

Since both p_1 and p_2 wait, we get that

$$\mathbb{E}[\Delta\,\mathrm{ALG}_t(p_1, p_2)] = 2d(p_1, p_2)(1 - e^{-\frac{t_2 - t_1}{d(p_1,p_2)}}) = 2\mathbb{E}[\Delta\,\mathrm{ALG}_d(p_1, p_2)].$$

Summing over all the pairs of tentatively matched requests and adding the delay cost of the unmatched requests, we get that

$$\mathbb{E}[\mathrm{ALG}_t(t_1, t_2)] \leq 2\mathbb{E}[\mathrm{ALG}_d(t_1, t_2)] + \mathrm{SOL}_t(t_1, t_2).$$

We conclude the proof of the lemma by summing over all the intervals, and by linearity of expectation, we get

$$\mathbb{E}[\mathrm{ALG}_t] \leq 2\mathbb{E}[\mathrm{ALG}_d] + \mathrm{SOL}_t\,.$$

□

C Proofs Omitted from Section 5

In order to prove Lemma 6, we need the following observation.

Observation 1. *Given a set of requests on the vertices of T which contains an equal number of positive and negative requests, let M be a minimum cost perfect matching between the positive and the negative requests (where, as usual, the cost of matching two requests is the distance between their locations under the tree metric). Then for any vertex v of the tree, the number of requests inside T_v that are matched in M to requests outside T_v is precisely $|\operatorname{sur}(v)|$. Furthermore, all these requests have the same sign as $\operatorname{sur}(v)$.*

Proof of Lemma 6. Let U^+ (resp. U^-) denote the set of positively (resp. negatively) unsaturated vertices v with $\operatorname{sur}(v) > 0$ (resp. $\operatorname{sur}(v) < 0$). Note that U^+ and U^- are disjoint. From the description of the algorithm, the rate of increase of $\sum_u (y_u^+ + y_u^-)$ is $\sum_{v \in U^+ \cup U^-} |\operatorname{sur}(v)|$. The rate of increase of the delay cost is equal to the number of pending requests. Thus, it is sufficient to prove that the number of pending requests is at most $2\sum_{v \in U^+ \cup U^-} |\operatorname{sur}(v)|$ at any time (except at instants when requests are connected).

Consider an arbitrary time instant. Recall that $\operatorname{sur}(r)$ is equal to the number of positive pending requests minus the number of negative pending requests. If $\operatorname{sur}(r) \neq 0$, augment the set of pending requests with $|\operatorname{sur}(r)|$ artificial requests located at r, with sign opposite to the sign of $\operatorname{sur}(r)$, resulting in a balanced set of requests. Let M be a minimum cost perfect matching between the positive and the negative requests in this set. First, consider the $|\operatorname{sur}(r)|$ pending requests that get matched to the $|\operatorname{sur}(r)|$ augmented requests at r. Charge these pending requests to r, and note that $r \in U^+ \cup U^-$ (unless $\operatorname{sur}(r) = 0$). Next, let (R^+, R^-) be a match in M, where R^+ (resp. R^-) is a positive (resp. negative) pending request located at u^+ (resp. u^-), and let $v = \operatorname{lca}(u^+, u^-)$. Since the algorithm has not connected R^+ and R^-, at least one of the following must be true.

1. There is a vertex $v' \neq v$ on the path from u^+ to v such that $e_{v'} \notin F^+$, i.e. v' is positively unsaturated.

2. There is a vertex $v' \neq v$ on the path from u^- to v such that $e_{v'} \notin F^-$, i.e. v' is negatively unsaturated.

Consider the first case. By Observation 1, since M matches the positive request $R^+ \in T_{v'}$ to $R^- \notin T_{v'}$, we have $\operatorname{sur}(v') > 0$. Additionally, since v' is positively unsaturated, $v' \in U^+$. By similar argument, in the second case, $v' \in U^-$. In either case, charge the pair of requests (R^+, R^-) to the vertex $v' \in U^+ \cup U^-$.[1] Observe that if this charging scheme charges a pair of pending requests to a vertex v, then one of the requests is in T_v, the other is outside T_v, and the pair is included in M. Again, by Observation 1, the number of pairs charged to any vertex v is at most $|\operatorname{sur}(v)|$. Thus, the number of pending requests is at most $2\sum_{v \in U^+ \cup U^-} |\operatorname{sur}(v)|$, as required. □

Proof of Lemma 7. We use the potential function technique. We design a potential function ϕ such that in each phase, the changes $\Delta(y_u^+ + y_u^-)$, $\Delta\phi$, and $\Delta(x_u + x'_u)$ satisfy

$$\Delta(y_u^+ + y_u^-) + \Delta\phi \leq 4\Delta(x_u + x'_u) \tag{1}$$

and $\phi = 0$ at the beginning as well as at the end of the algorithm's run. Summing (1) over all phases, we get the result.

Recall that $\operatorname{sur}^*(u)$ denotes the surplus of vertex u resulting from SOL. Define $\phi = 4d_u \cdot |\operatorname{sur}^*(u) - \operatorname{sur}(u)|$. Clearly, at the beginning as well as at the end, we have $\operatorname{sur}(u) = \operatorname{sur}^*(u) = 0$,

[1] If both cases hold, or if one of the cases holds for more than one v', then pick an arbitrary one.

and thus, $\phi = 0$. Observe that $\mathrm{sur}^*(u) - \mathrm{sur}(u)$ (and hence, ϕ) remains unchanged when new requests are given. The only events resulting in a change in $\mathrm{sur}^*(u) - \mathrm{sur}(u)$ are either SOL or the algorithm connecting a request inside T_u to one outside T_u. Also, x_u increases at a rate of at least $|\mathrm{sur}^*(u)|$.

In each phase of a vertex u, each of y_u^+ and y_u^- increases by at most $2d_u$, and therefore, $\Delta(y_u^+ + y_u^-) \leq 4d_u$. Except the last phase, in every phase, at least one of y_u^+ and y_u^- increases by exactly $2d_u$, and the phase ends with the algorithm connecting a request inside T_u to one outside T_u. We call such a phase *complete*, and we call the last phase *incomplete*. We prove that (1) holds first for complete phases, and then for the incomplete phase.

Let $k \geq 0$ denote the (absolute) number of requests in T_u which SOL connected to requests outside T_u during an arbitrary phase. Thus, $\Delta x'_u \geq kd_u$.

Consider any complete phase of vertex u and, without loss of generality, assume that the phase ends due to a positive request inside T_u getting connected to a negative request outside T_u. This means that z_u^+ increases from 0 to $2d_u$ in the phase. Since the only events resulting in a change in $\mathrm{sur}^*(u) - \mathrm{sur}(u)$ are either SOL or the algorithm connecting a request inside T_u to one outside, we have

$$\Delta|\,\mathrm{sur}^*(u) - \mathrm{sur}(u)| \leq |\Delta(\mathrm{sur}^*(u) - \mathrm{sur}(u))| \leq k + 1 \tag{2}$$

First, consider the case where $\Delta|\,\mathrm{sur}^*(u) - \mathrm{sur}(u)| = k + 1$, and therefore, $\Delta\phi = 4(k+1) \cdot d_u$. Now both inequalities in (2) are tight. Because the second inequality is tight, all the k requests inside T_u which SOL connected outside must be negative, and $\Delta(\mathrm{sur}^*(u) - \mathrm{sur}(u)) = k + 1 > 0$. Furthermore, $\mathrm{sur}^*(u) - \mathrm{sur}(u)$ never decreases during the phase. Because the first inequality in (2) is tight, the sign of $\mathrm{sur}^*(u) - \mathrm{sur}(u)$ at the beginning of the phase must be the same as that of $\Delta(\mathrm{sur}^*(u) - \mathrm{sur}(u))$, implying $\mathrm{sur}^*(u) - \mathrm{sur}(u) \geq 0$ initially. Since $\mathrm{sur}^*(u) - \mathrm{sur}(u)$ never decreases, we have $\mathrm{sur}^*(u) - \mathrm{sur}(u) \geq 0$ throughout the phase. Therefore, at any moment when z_u^+ was increasing, we have $\mathrm{sur}^*(u) \geq \mathrm{sur}(u) > 0$. Thus, the rate of increase of x_u is always at least as much as the rate of increase of z_u^+. Since z_u^+ increases by $2d_u$, we have $\Delta x_u \geq 2d_u$. Therefore,

$$\Delta(y_u^+ + y_u^-) + \Delta\phi \leq 4d_u + 4(k+1) \cdot d_u = 4(2d_u + kd_u) \leq 4\Delta(x_u + x'_u)$$

Next, suppose that $\Delta|\,\mathrm{sur}^*(u) - \mathrm{sur}(u)| < k + 1$. Observe that the parity of $\mathrm{sur}^*(u) - \mathrm{sur}(u)$ changes $k+1$ times during the phase: each time when the algorithm or SOL connects a request in T_u to one outside. Thus, if $\Delta|\,\mathrm{sur}^*(u) - \mathrm{sur}(u)|$ is not $k+1$, it must be at most $k-1$, which means $\Delta\phi \leq 4(k-1) \cdot d_u$. Therefore,

$$\Delta(y_u^+ + y_u^-) + \Delta\phi \leq 4d_u + 4(k-1) \cdot d_u = 4kd_u = 4\Delta x'_u \leq 4\Delta(x_u + x'_u)$$

Thus, in any case, (1) holds for any complete phase.

Finally, consider the last incomplete phase, which does not have a usage of e_u by the algorithm at the end. Note that at the end of the algorithm's run, $\mathrm{sur}(u) = \mathrm{sur}^*(u) = 0$, and hence, $\phi = 0$. Since ϕ is non-negative by definition, we have $\Delta\phi \leq 0$. If $k > 0$, then $\Delta(x_u + x'_u) \geq \Delta x'_u = kd_u \geq d_u$. Since $\Delta(y_u^+ + y_u^-) \leq 4d_u$, (1) holds. On the other hand, if $k = 0$, then $\mathrm{sur}^*(u) - \mathrm{sur}(u)$ stays constant in the phase. Since it is zero finally, it is zero throughout the phase. Thus, $\mathrm{sur}^*(u) = \mathrm{sur}(u)$ in the entire phase. Since $y_u^+ + y_u^-$ increases at a rate at most $|\mathrm{sur}(u)|$ and x_u increases at a rate at least $|\mathrm{sur}^*(u)|$, we have $\Delta(y_u^+ + y_u^-) \leq \Delta x_u$, again implying (1). □

D Proofs Omitted from Section 6

Proof of Lemma 9. For $j \in \{0, 1\}$, let M_j be the matching on the set $X \cup Y_j$ which achieves the cost $\mathrm{MIN}(X \cup Y_j, c)$. Consider the set of edges $M_0 \cup M_1$. This is a union of vertex-disjoint paths in

which every vertex in $Y_0 \cup Y_1$ has degree at most one. Thus, each path has all its vertices, except possibly the endpoints, in X.

Construct a matching M on $Y_0 \cup Y_1$ as follows. For each maximal path p in $M_0 \cup M_1$, do the following. If both endpoints of p are in X (which means that the whole path p is in X), ignore p. Else if p has length 0, that is, p is a single vertex from $Y_0 \cup Y_1$, leave it unmatched in M. Else, denote the endpoints of p by u and v. If both u and v are in $Y_0 \cup Y_1$, match u and v in M, and charge this cost to the weight of p. If $u \in X$ and $v \in Y_0 \cup Y_1$, leave v unmatched in M, and charge this cost to the cost of leaving u unmatched in one of the M_js. Thus, the contribution of every path to the cost $\mathrm{MIN}(X \cup Y_0, c) + \mathrm{MIN}(X \cup Y_1, c)$ is at least as much as its contribution to the cost of M. □

We now proceed to prove Lemma 11. For $x \in [0,1]$, define the *phase-i cumulative surplus* at x to be the signed total of the requests from phase i that are located in $[0,x]$, and denote it by $\mathrm{csur}_i(x)$. Then $\mathrm{csur}_i(x) \in \{0,1\}$ if $b_i = +1$, and $\mathrm{csur}_i(x) \in \{-1,0\}$ if $b_i = -1$. Define $\mathrm{csur}(x) = \sum_{i=0}^{r} \mathrm{csur}_i(x)$, the *cumulative surplus* at x, which is the signed total of all requests from all phases that are located in $[0,x]$. Observe that for any x, any feasible solution to the instance must connect at least $|\mathrm{csur}(x)|$ requests located to the left of x to the same number of requests located to the right of x. Hence, the connection cost of any feasible solution must be at least $\int_0^1 |\mathrm{csur}(x)|dx$. Moreover, there exists a solution, say SOL, whose connection cost is precisely $\int_0^1 |\mathrm{csur}(x)|dx$ (connect the t^{th} positive request and the t^{th} negative request from the left, for all t). This will be our adversarial solution to the instance. In order to bound the connection cost of SOL from above, we need prove that $\mathbb{E}_{b_0,\ldots,b_r,z_1,\ldots,z_r}[\int_0^1 |\mathrm{csur}(x)|dx]$ is small. We prove something stronger: we prove that the expectation is small enough even if we condition over the values of $z_1,\ldots,z_r$, and only average over $b_0,\ldots,b_r$.

Lemma 13. *For every fixed* $(z_1,\ldots,z_r) \in \{0,1\}^r$, $\mathbb{E}_{b_0,\ldots,b_r}\left[\int_0^1 |\mathrm{csur}(x)|dx\right] = O(\sqrt{r})$.

Proof. Since $\mathbb{E}_{b_0,\ldots,b_r}\left[\int_0^1 |\mathrm{csur}(x)|dx\right] = \int_0^1 \mathbb{E}_{b_0,\ldots,b_r}\left[|\mathrm{csur}(x)|\right] dx$, it is sufficient to prove that $\mathbb{E}\left[|\mathrm{csur}(x)|\right] = O(\sqrt{r})$.

Given $z_1,\ldots,z_r$, the locations of the requests are fixed. Observe that $\mathrm{csur}_i(x)$ is zero if the number of requests of phase i in $[0,x]$ is even. If that number is odd, then $\mathrm{csur}_i(x) = b_i$ is $+1$ and -1 with probability $1/2$ each. Thus, $\mathrm{csur}(x) = \sum_{i=0}^{r} \mathrm{csur}_i(x)$ is the sum of at most $r+1$ independent random variables, each of which takes values $+1$ and -1 with equal probability, where the number of random variables is determined by x and $z_1,\ldots,z_r$. Therefore $|\mathrm{csur}(x)|$ is the deviation of a random walk of at most $r+1$ steps on the integers starting from 0, and moving in either direction with equal probability. Using a standard result,[1] we have $\mathbb{E}\left[|\mathrm{csur}(x)|\right] = O(\sqrt{r})$, as required. □

Taking the solution SOL which minimizes the connection cost as the adversarial solution, we now prove an upper bound on the expected cost of the optimum solution of a random MBPMD instance drawn from $\mathcal{D}$.

Proof of Lemma 11. Consider the solution SOL. By Lemma 13, its expected connection cost is made $O(\sqrt{r}) = O(\sqrt{L/\log L})$, and we are left to bound its expected delay cost. Note that the sum of the arrival times of all requests in an instance is an upper bound on the delay cost of every solution to the instance (which keeps a request waiting only until its partner arrives). In particular, this applies to SOL. For instances in the support of $\mathcal{D}$, the sum of the arrival times is the same, and

[1] For instance: http://mathworld.wolfram.com/RandomWalk1-Dimensional.html

is equal to $\sum_{i=0}^{r} n_i \sum_{j=0}^{i-1} t_j$, where $n_i = 2L^{r-i}$ is the number of requests in phase i, and $t_j = 1/L^{r-j}$ is the duration of phase j. Thus, the delay cost is bounded from above by

$$\sum_{i=0}^{r} n_i \sum_{j=0}^{i-1} t_j = \sum_{i=0}^{r} 2L^{r-i} \sum_{j=0}^{i-1} \frac{1}{L^{r-j}} = 2 \sum_{i=0}^{r} \frac{1}{L^i} \sum_{j=0}^{i-1} L^j = 2 \sum_{i=0}^{r} \frac{1}{L^i} \cdot \frac{L^i - 1}{L - 1} \leq \frac{2r}{L-1}$$

which is $O(1/\log L)$, since $r = \lfloor L/\log L \rfloor$. Thus, the expected cost of a random MBPMD instance drawn from $\mathcal{D}$ is $O(\sqrt{L/\log L}) + O(1/\log L) = O(\sqrt{L/\log L})$. □

Proof of Lemma 12. Construct a solution as follows. For i decreasing from r to 1, connect each unmatched request from phase i to the request from phase $i-1$ located at the same point. This is possible because $S_i \subseteq S_{i-1}$, and results in zero connection cost. The delay cost is at most the number of requests in phase i times the duration of phase $i-1$, which is $2L^{r-i} \cdot 1/L^{r-i+1} = 2/L$. Finally pair up the unmatched requests from phase 0 optimally, with connection cost at most 1. The overall connection cost is 1, and the overall delay cost is 2/L for each phase except phase 0. Thus, the total cost of the solution is $1 + 2r/L \leq 1 + 2/\log_2 L$. □