Optimal Analysis of an Online Algorithm for the Bipartite Matching Problem on a Line

Sharath Raghvendra
Virginia Tech
Blacksburg, USA
sharathr@vt.edu

Abstract

In the online metric bipartite matching problem, we are given a set S of server locations in a metric space. Requests arrive one at a time, and on its arrival, we need to immediately and irrevocably match it to a server at a cost which is equal to the distance between these locations. A α-competitive algorithm will assign requests to servers so that the total cost is at most α times the cost of M_{OPT} where M_{OPT} is the minimum cost matching between S and R.

We consider this problem in the adversarial model for the case where S and R are points on a line and $|S| = |R| = n$. We improve the analysis of the deterministic Robust Matching Algorithm (RM-Algorithm, Nayyar and Raghvendra FOCS'17) from $O(\log^2 n)$ to an optimal $\Theta(\log n)$. Previously, only a randomized algorithm under a weaker oblivious adversary achieved a competitive ratio of $O(\log n)$ (Gupta and Lewi, ICALP'12). The well-known Work Function Algorithm (WFA) has a competitive ratio of $O(n)$ and $\Omega(\log n)$ for this problem. Therefore, WFA cannot achieve an asymptotically better competitive ratio than the RM-Algorithm.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Bipartite Matching, Online Algorithms, Adversarial Model, Line Metric

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.67

Related Version A full version of this paper is available at `https://arxiv.org/abs/1803.07206`.

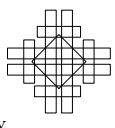
Funding This work is supported by a NSF CRII grant NSF-CCF 1464276

1 Introduction

Driven by consumers' demand for quick access to goods and services, business ventures schedule their delivery in real-time, often without the complete knowledge of the future request locations or their order of arrival. Due to this lack of complete information, decisions made tend to be sub-optimal. Therefore, there is a need for competitive *online algorithms* which immediately and irrevocably allocate resources to requests in real-time by incurring a small cost.

Motivated by these real-time delivery problems, we study the problem of computing the online metric bipartite matching of requests to servers. Consider servers placed in a metric space where each server has a capacity that restricts how many requests it can serve. When a new request arrives, one of the servers with positive capacity is matched to this request. After this request is served, the capacity of the server reduces by one. We assume that the cost associated with this assignment is a metric cost; for instance, it could be the minimum distance traveled by the server to reach the request.

The case where the capacity of every server is ∞ is the celebrated k-server problem. The case where every server has a capacity of 1 is the *online metric bipartite matching problem.*




In this case, the requests arrive one at a time, we have to immediately and irrevocably match it to some unmatched server. We assume there are n servers and n requests. The resulting assignment is a matching and is referred to as an *online matching*. An optimal assignment is impossible since an adversary can easily fill up the remaining locations of requests in R in a way that our current assignment becomes sub-optimal. Therefore, we want our algorithm to compute an assignment that is competitive with respect to the optimal matching. For any input S, R and any arrival order of requests in R, we say our algorithm is α-*competitive*, for $\alpha > 1$, when the cost of the online matching M is at most α times the minimum cost, i.e.,

$$w(M) \leq \alpha w(M_{\text{OPT}}).$$

Here M_{OPT} is the minimum-cost matching of the locations in S and R. In the above discussion, note the role of the adversary. In the *adversarial model*, the adversary knows the server locations and the assignments made by the algorithm and generates a sequence to maximize α. In the weaker oblivious adversary model, the adversary knows the randomized algorithm but does not know the random choices made by the algorithm. In this paper, we consider the online metric bipartite matching problem in the adversarial model and where S and R are points on a line.

Consider the adversarial model. For any request, the *greedy heuristic* simply assigns the closest unmatched server to it. The greedy heuristic, even for the line metric, is only 2^n-competitive [3] for the online matching problem. The well-known Work Function Algorithm (WFA) chooses a server that minimizes the sum of the *greedy cost* and the so-called *retrospective cost*. For the k-server problem, the competitive ratio of the WFA is $2k - 1$ [6] which is near optimal with a lower bound of k on the competitive ratio of any algorithm in any metric space that has at least $k + 1$ points [7].

In the context of online metric matching problem, there are algorithms that achieve a competitive ratio of $2n - 1$ in the adversarial model [9, 5, 4]. This competitive ratio is worst-case optimal, i.e., there exists a metric space where we cannot do better. However, for Euclidean metric, for a long time, there was a stark contrast between the upper bound of $2n - 1$ and the lower bound of $\Omega(n^{1-1/d})$. Consequently, significant effort has been made to study the performance of online algorithms in special metric spaces, especially the line metric. For example, for the line metric, it has been shown that the WFA when applied to the online matching problem has a lower bound of $\Omega(\log n)$ and an upper bound of $O(n)$ [2]; see also [1] for a $O(n^{0.585})$competitive algorithm for the line metric. In the oblivious adversary model, there is an $O(\log n)$-competitive [3] algorithm for the line metric. There is also an $O(d \log n)$-competitive algorithm in the oblivious adversary model for any metric space with doubling dimension d [3].

Only recently, for any metric space and for the adversarial model, Raghvendra and Nayyar [8] provided a bound of $O(\mu(S) \log^2 n)$ on the competitive ratio of the RM-Algorithm – an algorithm that was introduced by Raghvendra [9]; here $\mu(S)$ is the worst case ratio of the the TSP and diameter of any positive diameter subset of S. There is a simple lower bound of $\Omega(\mu(S))$ on the competitive ratio of any algorithm for this problem. Therefore, RM-Algorithm is near-optimal for every input set S. When S is a set of points on a line, $\mu(S) = 2$ and therefore, their analysis bounds the competitive ratio of the RM-Algorithm by $O(\log^2 n)$ for the line metric and $O(n^{1-1/d} \log^2 n)$ for any d-dimensional Euclidean metric. Furthermore, RM-Algorithm also has a lower bound of $\Omega(\log n)$ on its competitive ratio for the line metric. In this paper, we provide a different analysis and show that the RM-Algorithm is $\Theta(\log n)$-competitive.

Overview of RM-Algorithm. At a high-level, the RM-Algorithm maintains two matchings M and M^*, both of which match requests seen so far to the same subset of servers in S. We refer to M as the online matching and M^* as the offline matching. For a parameter $t > 1$ chosen by the algorithm, the offline matching M^* is a t-approximate minimum-cost matching satisfying a set of relaxed dual feasibility conditions of the linear program for the minimum-cost matching; here each constraint relaxed by a multiplicative factor t. Note that, when $t = 1$, the matching M^* is simply the minimum-cost matching.

When the i^{th} request r_i arrives, the algorithm computes an augmenting path P_i with the minimum "cost" for an appropriately defined cost function. This path P_i starts at r_i and ends at an unmatched server s_i. The algorithm then augments the offline matching M^* along P whereas the online matching M will simply match the two endpoints of P_i. Note that M and M^* will always match requests to the same subset of servers. We refer to the steps taken by the algorithm to process request r_i as the i^{th} phase of the algorithm. For $t > 1$, it has been shown in [9] that the sum of the costs of every augmenting path computed by the algorithm is bounds the online matching cost from above. Nayyar and Raghvendra [8] use this property and bounded the ratio of the sum of the costs of augmenting paths to the cost of the optimal matching. In order to accomplish this they associate every request r_i to the cost of P_i. To bound the sum of costs of all the augmenting paths, they partition the requests into $\Theta(\log^2 n)$ groups and within each group they bound this ratio by $\mu(S)$ (which is a constant when S is a set of points on a line). For the line metric, each group can, in the worst-case, have a ratio of $\Theta(1)$. However, not all groups can simultaneously exhibit this worst-case behavior. In order to improve the competitive ratio from $O(\log^2 n)$ to $O(\log n)$, therefore, one has to bound the combined ratios of several groups by a constant making the analysis challenging.

1.1 Our results and techniques

In this paper, we show that when the points in $S \cup R$ are on a line, the RM-Algorithm achieves a competitive ratio of $O(\log n)$. Our analysis is tight as there is an example in the line metric for which the RM-Algorithm produces an online matching which is $\Theta(\log n)$ times the optimal cost. We achieve this improvement using the following new ideas:

- First, we establish the the *ANFS-property* of the RM-Algorithm (Section 3.1). We show that many requests are matched to an approximately nearest free server (ANFS) by the algorithm. We define certain edges of the online matching M as *short* edges and show that every short edge matches the request to an approximately nearest free server. Let M_H be the set of short edges of the online matching and $M_L = M \setminus M_H$ be the *long* edges. We also show that when $t = 3$, the total cost of the short edges $w(M_H) \geq w(M)/6$ and therefore, the cost of the long edges is $w(M_L) < (5/6)w(M)$.
- For every point in $S \cup R$, the RM-Algorithm maintains a dual weight (Section 2). For our analysis in the line metric, we assign an interval to every request. The length of this interval is determined by the dual weight of the request. At the start of phase i, let σ_{i-1} be the union of all such intervals. By its construction, σ_{i-1} will consist of a set of interior-disjoint intervals. While processing request r_i, the RM-Algorithm conducts a series of dual adjustments and a subset of requests (referred to as B_i) undergo an increase in dual weights. After the dual adjustments, the union of the intervals for requests in B_i forms a single interval and these increases conclude in the discovery of the minimum cost augmenting path. Therefore, after phase i, intervals in σ_{i-1} may grow and combine to form a single interval $\mathcal{I}$ in σ_i. Furthermore, the new online edge (s_i, r_i) is also contained inside this newly formed interval $\mathcal{I}$ (Section 3.3). Based on the length of the interval $\mathcal{I}$, we assign one of $O(\log n)$ levels to the edge (s_i, r_i). This partitions all the online edges in $O(\log n)$ levels.

- The online edges of any given level k can be expressed as several non-overlapping well-aligned matching of a well separated input (Section 4) . We establish properties of such matchings (Section 3.2 and Figure 1) and use it to bound the total cost of the "short" online edges of level k by the sum of $w(M_{\text{OPT}})$ and γ times the cost of the long edges of level k, where γ is a small positive constant (Section 4). Adding across the $O(\log n)$ levels, we get $w(M_H) \leq O(\log n)w(M_{\text{OPT}}) + \gamma w(M_L)$. Using the ANFS-property of short and long edges, we immediately get $(1/6 - 5\gamma/6)w(M) \leq O(\log n)w(M_{\text{OPT}})$. For a sufficiently small γ, we bound the competitive ratio, i.e., $w(M)/w(M_{\text{OPT}})$ by $O(\log n)$.

Organization. The rest of the paper is organized as follows. We begin by presenting (in Section 2) the RM-Algorithm and some of its use properties as shown in [9]. For the analysis, we establish the *ANFS-property* in Section 3.1. After that, we will (in Section 3.2) introduce well aligned matchings of well-separated inputs on a line. Then, in Section 3.3, we interpret the dual weight maintained for each request as an interval and study the properties of the union of these intervals. Using these properties (in Section 4) along with the ANFS-property of the algorithm, we will establish a bound of $O(\log n)$ on the competitive ratio for the line metric.

2 Background and algorithm details

In this section, we introduce the relevant background and describe the RM-algorithm.

A *matching* $M \subseteq S \times R$ is any set of vertex-disjoint edges of the complete bipartite graph denoted by $G(S, R)$. The cost of any edge $(s, r) \in S \times R$ is given by $d(s, r)$; we assume that the cost function satisfies the metric property. The cost of any matching M is given by the sum of the costs of its edges, i.e., $w(M) = \sum_{(s,r)\in M} d(s, r)$. A *perfect matching* is a matching where every server in S is serving exactly one request in R, i.e., $|M| = n$. A *minimum-cost perfect matching* is a perfect matching with the smallest cost.

Given a matching M^*, an *alternating path* (resp. cycle) is a simple path (resp. cycle) whose edges alternate between those in M^* and those not in M^*. We refer to any vertex that is not matched in M^* as a *free vertex*. An *augmenting path* P is an alternating path between two free vertices. We can *augment* M^* by one edge along P if we remove the edges of $P \cap M^*$ from M^* and add the edges of $P \setminus M^*$ to M^*. After augmenting, the new matching is precisely given by $M^* \oplus P$, where $\oplus$ is the symmetric difference operator. A matching M^* and a set of dual weights, denoted by $y(v)$ for each point $v \in S \cup R$, is a *t-feasible matching* if, for any request $r \in R$ and a server $s \in S$, the following conditions hold:

$$y(s) + y(r) \leq td(s, r), \tag{1}$$

$$y(s) + y(r) = d(s, r) \quad \text{if } (s, r) \in M^*. \tag{2}$$

Also, we refer to an edge $(s, r) \in S \times R$ to be *eligible* if either $(s, r) \in M^*$ or (s, r) satisfies inequality (1) with equality:

$$y(s) + y(r) = td(s, r), \quad \text{if } (s, r) \notin M^* \tag{3}$$

$$y(s) + y(r) = d(s, r) \quad \text{if } (s, r) \in M^*. \tag{4}$$

For a parameter $t \geq 1$, we define the *t-net-cost* of any augmenting path P with respect to M^* to be:

$$\phi_t(P) = t \left(\sum_{(s,r)\in P\setminus M^*} d(s, r) \right) - \sum_{(s,r)\in P\cap M^*} d(s, r).$$

The definitions of t-feasible matching, eligible edges and t-net cost (when $t = 1$) are also used in describing the well-known Hungarian algorithm which computes the minimum-cost matching. In the Hungarian algorithm, initially $M^* = \emptyset$ is a 1-feasible matching with all the dual weights $y(v)$ set to 0. In each iteration, the Hungarian Search procedure adjusts the dual weights $y(\cdot)$ and computes an augmenting path P of eligible edges while maintaining the 1-feasibility of M^* and then augments M^* along P. The augmenting path P computed by the standard implementation of the Hungarian search procedure can be shown to also have the minimum 1-net-cost.

Using this background, we describe the RM-Algorithm. At the start, the value of t is chosen at the start of the algorithm. The algorithm maintains two matchings: an online matching M and a t-feasible matching (also called the *offline matching*) M^* both of which are initialized to $\emptyset$. After processing $i-1$ requests, both matchings M and M^* match each of the $i-1$ requests to the same subset of servers in S, i.e., the set of free (unmatched) servers S_F is the same for both M and M^*. To process the i^{th} request r_i, the algorithm does the following

1. Compute the minimum t-net-cost augmenting path P_i with respect to the offline matching M^*. Let P_i be this path starting from r_i and ending at some server $s_i \in S_F$.
2. Update offline matching M^* by augmenting it along P_i, i.e., $M^* \leftarrow M^* \oplus P_i$ and update online matching M by matching r_i to s_i. $M \leftarrow M \cup \{(s_i, r_i)\}$.

While computing the minimum t-net-cost augmenting path in Step 1, if there are multiple paths with the same t-net-cost, the algorithm will simply select the one with the fewest number of edges. Throughout this paper, we set $t = 3$. In [9], we present an $O(n^2)$-time search algorithm that is similar to the Hungarian Search to compute the minimum t-net-cost path in Step 1 any phase i and we describe this next.

The implementation of Step 1 of the algorithm is similar in style to the Hungarian Search procedure. To compute the minimum t-net-cost path P_i , the algorithm grows an alternating tree consisting only of eligible edges. There is an alternating path of eligible edges from r_i to every server and request participating in this tree. To grow this tree, the algorithm increases the dual weights of every request in this tree until at least one more edge becomes eligible and a new vertex enters the tree. In order to maintain feasibility, the algorithm reduces the dual weights of all the servers in this tree by the same amount. This search procedure ends when an augmenting path P_i consisting only of eligible edges is found. Let B_i (resp. A_i) be the set of requests (resp. servers) that participated in the alternating tree of phase i. Note that during Step 1, the dual weights of requests in B_i may only increase and the dual weights of servers in A_i may only reduce.

The second step begins once the augmenting path P_i is found. The algorithm augments the offline matching M^* along this path. Note that, for the M^* to be t-feasible, the edges that newly enter M^* must satisfy (2). In order to ensure this, the algorithm will reduce the dual weight of each request r on P_i to $y(r) \leftarrow y(r) - (t-1)d(s,r)$. Further details of the algorithm and proof of its correctness can be found in [9]. In addition, it has also been shown that the algorithm maintains the following three invariants:

(I1) The offline matching M^* and dual weights $y(\cdot)$ form a t-feasible matching,

(I2) For every server $s \in S$, $y(s) \leq 0$ and if $s \in S_F$, $y(s) = 0$. For every request $r \in R$, $y(r) \geq 0$ and if r has not yet arrived, $y(r) = 0$,

(I3) At the end of the first step of phase i of the algorithm the augmenting path P_i is found and the dual weight of r_i, $y(r_i)$, is equal to the t-net-cost $\phi_t(P_i)$.

Notations. Throughout the rest of this paper, we will use the following notations. We will index the requests in their order of arrival, i.e., r_i is the ith request to arrive. Let R_i be the set of first i request. Our algorithm processes the request r_i, by computing an augmenting path P_i. Let s_i be the free server at the other end of the augmenting path P_i. Let $\mathbb{P} = \{P_1, \ldots, P_n\}$ be the set of n augmenting paths generated by the algorithm. In order to compute the augmenting path P_i, in the first step, the algorithm adjusts the dual weights and constructs an alternating tree; let B_i be the set of requests and let A_i be the set of servers that participate in this alternating tree. Let M_i^* be the offline matching after the ith request has been processed; i.e., the matching obtained after augmenting the matching M_{i-1}^* along P_i. Note that $M_0^* = \emptyset$ and $M_n^* = M^*$ is the final matching after all the n requests have been processed. The online matching M_i is the online matching after i requests have been processed. M_i consists of edges $\bigcup_{j=1}^{i}(s_j, r_j)$. Let S_F^i be the free servers with respect to matchings M_{i-1} and M_{i-1}^*, i.e., the set of free servers at the start of phase i. For any path P, let $\ell(P) = \sum_{(s,r)\in P} d(s,r)$ be its *length*.

Next, in Section 3.1, we will present the approximate nearest free server (ANFS) property of the RM-Algorithm. In Section 3.2, we present an well aligned matching of a well separated input instance. In Section 3.3, we interpret the execution of each phase of the RM-Algorithm in the line metric. Finally, in Section 4, we give our analysis of the algorithm for the line metric.

3 New properties of the algorithm

In this section, we present new properties of the RM-Algorithm. First, we show that the RM-Algorithm will assign an approximate nearest free server for many requests and show that the total cost of these "short" matches will be at least one-sixth of the online matching cost. Our proof of this property is valid for any metric space.

3.1 Approximate nearest free server property

We divide the n augmenting paths $\mathbb{P} = \{P_1, \ldots, P_n\}$ computed by the RM-Algorithm into two sets, namely *short* and *long* paths. For any i, we refer to P_i as a *short* augmenting path if $\ell(P_i) \leq \frac{4}{t-1}\phi_t(P_i)$ and *long* otherwise. Let $H \subseteq \mathbb{P}$ be this set of all short augmenting paths and $L = \mathbb{P} \setminus H$ be the *long* augmenting paths. In phase i, the algorithm adds an edge between s_i and r_i in the online matching. We refer to any edge of the online matching (s_i, r_i) as a *short edge* if P_i is a short augmenting path. Otherwise, we refer to this edge as a *long* edge. The set of all short edges, M_H and the set of long edges M_L partition the edges of the online matching M.

At the start of phase i, S_F^i are the set of free servers. Let $s^* \in S_F^i$ be the server closest to r_i, i.e., $s^* = \arg\min_{s\in S_F^i} d(r_i, s)$. Any other server $s \in S_F^i$ is an *α-approximate nearest free server* to the request r if

$$d(r_i, s) \leq \alpha d(r_i, s^*).$$

In Lemma 1 and 2, we show that the short edges in M_H match a request to a 6-ANFS and the cost of $w(M_H)$ is at least one-sixth of the cost of the online matching.

▶ **Lemma 1.** *For any request r_i, if P_i is a short augmenting path, then s_i is a $(4+\frac{4}{t-1})$-ANFS of r_i.*

Proof. Let s^* be the nearest available server of r_i in S_F^i. Both s^* and r_i are free and so the edge $P = (s^*, r_i)$ is also an augmenting path with respect to M_{i-1}^* with $\phi_t(P) = td(s^*, r_i)$.

The algorithm computes P_i which is the minimum t-net-cost path with respect to M^*_{i-1} and so,

$$\phi_t(P_i) \leq td(s^*, r_i).$$

Since P_i is a short augmenting path,

$$\frac{(t-1)}{4} d(s_i, r_i) \leq \phi_t(P_i) \leq td(s^*, r_i), \tag{5}$$

$$d(s_i, r_i) \leq \frac{4t}{t-1} d(s^*, r_i) = (4 + \frac{4}{t-1}) d(s^*, r_i), \tag{6}$$

implying that s_i is a $(4 + \frac{4}{t-1})$-approximate nearest free server to the request r_i. ◀

▶ **Lemma 2.** *Let M_H be the set of short edges of the online matching M. Then,*

$$(4 + \frac{4}{t-1}) w(M_H) \geq w(M). \tag{7}$$

If we set $t = 3$, then $6w(M_H) \geq w(M)$ or $w(M_H) \geq w(M)/6$.

Convention and notations for the line metric. When S and R are points on a line, any point $v \in S \cup R$ is simply a real number. We can interpret any edge $(s, r) \in S \times R$ as the line segment that connects s and r and its cost $d(s, r)$ as $|s - r|$. We will abuse notation and use (s, r) to denote both the edge as well as the corresponding line segment. A matching of S and R, therefore, is also a set of line segments, one corresponding to each edge. For any closed interval $I = [x, y]$, $open(I)$ will be the open interval (x, y). We define the boundary, $bd(I)$ of any closed (or open) interval $I = [x, y]$ (or $I = (x, y)$) to be the set of two end points$\{x, y\}$. Any edge (s, r) of the matching is called a *left edge* if the server s is to the left of request r. Otherwise, we refer to the edge as the *right edge*.

3.2 Properties of 1-dimensional matching

In this section, we define certain input instances that we refer to as a well-separated input instance. We then define matchings that are well-aligned for such instance and then bound the cost of such a well-aligned matching by the cost of their optimal matching. In Section 4, we divide the edges of the online matching into well-aligned matchings on well-separated instances. This will play a critical role in bounding the competitive ratio for the line metric.

Well-separated instances and well aligned matchings. A *well-separated instance* of the 1-dimensional matching problem is defined as follows. For any $\Delta > 0$ and $0 < \varepsilon \leq 1/8$, consider the following four intervals $I_M = [0, \Delta]$, $I_L = [-\varepsilon\Delta, 0], I_R = [\Delta, (1 + \varepsilon)\Delta]$ and $I_A = [-\varepsilon\Delta, (1 + \varepsilon)\Delta]$. Note that I_L is the leftmost interval, I_R is the rightmost interval, and I_M lies in the middle of the I_L and I_R. I_A is simply the union of I_L, I_R, and I_M. We say that any input set of servers S and requests R is an ε-well-separated input if, for some $\Delta > 0$, there is a translation of $S \cup R$ such that $S \subset I_L \cup I_R$ and $R \subset I_A$.

Given an ε-well-separated input S and R, consider the intervals, $I'_L = [-\varepsilon\Delta, \varepsilon\Delta]$ and $I'_R = [(1 - \varepsilon)\Delta, (1 + \varepsilon)\Delta]$. We divide the edges of any matching M of S and R into three groups. Any edge $(s, r) \in M$ is *close* if $s, r \in I'_L$ or $s, r \in I'_R$. Any edge $(s, r) \in M$ is a *far* edge if $(s, r) \in I'_L \times I'_R$ or $(s, r) \in I'_R \times I'_L$. Let M_{close} and M_{far} denote the close and far edges of M respectively. For a matching edge (s, r), we denote it as a *medium edge* if the request r is inside the interval $[\varepsilon\Delta, (1 - \varepsilon\Delta)]$ and the server s is inside the interval I_L or

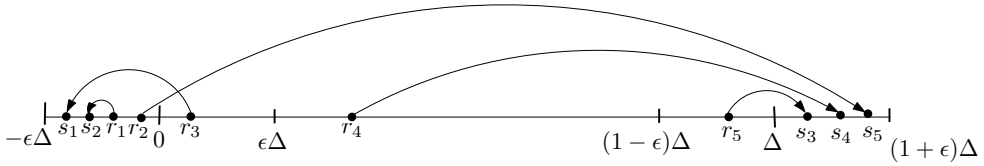


Figure 1 The server set is $S = \{s_1, s_2, s_3, s_4, s_5\} \subset [-\varepsilon\Delta, 0] \cup [\Delta, (1+\varepsilon)\Delta]$ and request set is $R = \{r_1, r_2, r_3, r_4, r_5\} \subset [-\varepsilon\Delta, (1+\varepsilon)\Delta]$. So, $S \cup R$ is an ε-well separated input instance. The matching $M = \{(r_1, s_2), (r_2, s_5), (r_3, s_1), (r_4, s_4), (r_5, s_3)\}$ is partitioned into $M_{\text{close}} = \{(r_1, s_2), (r_3, s_1), (r_5, s_3)\}$, $M_{\text{far}} = \{(r_2, s_5)\}$ and $M_{\text{med}} = \{(r_4, s_4)\}$. M is ε-well aligned since the edges of M_{close} in $[-\varepsilon\Delta, (1+\varepsilon)\Delta]$ are left edges (server is to the left of request) and those in $[(1-\varepsilon)\Delta, (1+\varepsilon)\Delta]$ are right edges (server is to the right of request).

I_R. We denote this set of edges as the *medium edges* and denote it by M_{med}. From the well-separated property of the input it is easy to see that $M = M_{\text{far}} \cup M_{\text{med}} \cup M_{\text{close}}$. A matching M is *ε-well-aligned* if all the edges of M_{close} with both their endpoints inside I'_R (resp. I'_L) are right (resp. left) edges. See Figure 1 for an example of ε-well aligned matching of an ε-well separated input instance. Any ε-well-aligned matching of an ε-well-separated input instance satisfies the following property.

▶ **Lemma 3.** *For any $0 \le \varepsilon \le 1/8$, given an ε-well-separated input S and R and an ε-well-aligned matching M, let M_{OPT} be the optimal matching of S and R and let $M_{\text{close}}, M_{\text{far}}$ and M_{med} be as defined above. Then,*

$$w(M_{\text{close}}) + w(M_{\text{med}}) \le \left(\frac{2}{\varepsilon} + 3\right) w(M_{\text{OPT}}) + \frac{4\varepsilon}{1-2\varepsilon} w(M_{\text{far}}).$$

3.3 Interpreting dual weights for the line metric

Next, we interpret the dual weights and their changes during the RM-Algorithm for the line metric and derive some of its useful properties.

Span of a request. For any request $r \in R$, let $y^i_{max}(r)$ be the largest dual weight that is assigned to r in the first i phases. The second step of phase i does not increase the dual weights of requests, and so, $y^i_{max}(r)$ must be a dual weight assigned at the end of first step of some phase $j \le i$. For any request r and any phase i, the span of r, denoted by $span(r, i)$, is an open interval that is centered at r and has a length of $\frac{2y^i_{max}(r)}{t}$, i.e., $span(r, i) = \left(r - \frac{y^i_{max}(r)}{t}, r + \frac{y^i_{max}(r)}{t}\right)$. We will refer to the closed interval $[r - \frac{y^i_{max}(r)}{t}, r + \frac{y^i_{max}(r)}{t}]$ with center r and length $\frac{2y^i_{max}(r)}{t}$ as $cspan(r, i)$.

Intuitively, request r may participate in one or more alternating trees in the first i phases of the algorithm. The dual weight of every request that participates in an alternating tree, including r, increases. These increases reflect their combined search for a free server. The region $span(r, i)$ represents the region swept by r in its search for a free server in the first i phases. We show in Lemma 4 that the span of any request does not contain a free server in its interior. We show that, during the search, if the span of a request r expands to include a free server $s \in S_F$ on its boundary, i.e., $s \in bd(span(r, i))$, then the algorithm would have found a minimum t-net-cost path and the search would stop. Therefore, the open interval $span(r, i)$ will remain empty.

▶ **Lemma 4.** *For every r, $span(r, i) \cap S^i_F = \emptyset$.*

Proof. For the sake of contradiction, we assume that the span of a request r contains a free server s, i.e., $s \in S_F^i \cap span(r,i)$. So, the distance between r and s is $|s-r| < \frac{y^i_{max}(r)}{t}$, or,

$$y^i_{max}(r) > t(|s-r|). \tag{8}$$

Since $y^i_{max}(r)$ is the largest dual weight assigned to r, there is a phase $j \leq i$, when request r is assigned this dual weight by the algorithm. Since the first step of the algorithm may only increase and the second step may only decrease the dual weights of any request, we can assume that $y(r)$ was assigned the dual weight of $y^i_{max}(r)$ at the end of the first step of some phase j. Let $y(s)$ and $y(r) = y^i_{max}(r)$ be the dual weights at the end of the first step of phase j. From Invariant (I1), it follows that $y^i_{max}(r) + y(s) \leq t(|s-r|)$. From this and (8), we have $y(s) < 0$. The free server s has $y(s) < 0$ contradicting invariant (I2). ◀

▶ **Lemma 5.** *Let (s,r) be any eligible edge at the end of the first step of phase i. Then,*

$$y^i_{max}(r) \geq t|s-r|,$$

implying that $s \in cspan(r,i)$ and the edge $(s,r) \subset span(r,i)$.

Proof. An edge is eligible if it is in M^*_{i-1} or if it satisfies (3). Suppose $(s,r) \notin M^*_{i-1}$ and satisfies (3). In this case, $y(s)+y(r) = t(|s-r|)$. From (I2), $y(s) \leq 0$ and so, $y(r) \geq t(|s-r|)$, implying that $y^i_{max}(r) \geq t(|s-r|)$.

For the case where $(s,r) \in M^*_{i-1}$, let $0 < j < i$ be the largest index such that $(s,r) \in M^*_j$ and $(s,r) \notin M^*_{j-1}$. Therefore, $(s,r) \in P_j \setminus M^*_{j-1}$. Since P_j is an augmenting path with respect to M^*_{j-1}, every edge of $P_j \setminus M^*_{j-1}$ satisfies the eligibility condition (4) at the end of the first step of phase j of the algorithm. For any vertex v, let $y'(v)$ be its dual weight after the end of the first step of phase j of the algorithm. From (4), we have $y'(r)+y'(s) \leq t|s-r|$. From (I2), since $y'(s) \leq 0$, we have $y'(r) \geq t|s-r|$. By definition, $y^i_{max}(r) \geq y'(r)$ and therefore $y^i_{max}(r) \geq t|s-r|$. ◀

Search interval of a request. Recollect that B_i is the set of requests that participate in the alternating tree of phase i. In Lemma 6, we show that $\bigcup_{r \in B_i} cspan(r,i)$ is a single closed interval. We define *search interval* of r_i, denoted by $sr(r_i)$, as the open interval $open(\bigcup_{r \in B_i} cspan(r,i))$. The search interval of a request represents the region searched for a free server by all the requests of B_i. In Lemma 6, we establish a useful property of search interval of a request. We show that the search interval of r_i does not contain any free servers of S^i_F and the free server s_i (the match of r_i in the online matching) is at the boundary of $sr(r_i)$. Since the search interval contains r_i, it follows that s_i is either the closest free server to the left or the closest free server to the right of r_i. Using the fact that all requests of B_i are connected to r_i by an path of eligible edges along with Lemma 4 and Lemma 5, we get the proof for Lemma 6.

▶ **Lemma 6.** *After phase i, $\bigcup_{r \in B_i} cspan(r,i)$ is a single closed interval and so, the search interval $sr(r_i)$ of any request r_i is a single open interval. Furthermore,*

- *All edges between A_i and B_i are inside the search interval of r_i, i.e.,$A_i \times B_i \subseteq sr(r_i)$,*
- *There are no free servers inside the search interval of r_i, i.e.,$S^i_F \cap sr(r_i) = \emptyset$, and,*
- *The server s_i chosen by the algorithm is on the boundary of the search interval of r_i, i.e., $s_i \in bd(sr(r_i))$.*

Cumulative search region. After phase i, the *cumulative search region* csr_i for the first i requests R_i is the union of the individual search intervals $\text{csr}_i = (sr(r_1) \cup sr(r_2) \ldots \cup sr(r_i))$. Since the cumulative search region is the union of open intervals, it is formed by a set of pairwise interior-disjoint open intervals. Let σ_i of csr_i be a sequence of these interior-disjoint open intervals in the cumulative search region ordered from left to right, i.e., $\sigma_i = \langle \mathcal{C}^i_1, \ldots, \mathcal{C}^i_k \rangle$, where $\mathcal{C}^i_j$ appears before $\mathcal{C}^i_l$ in the sequence if and only if the interval $\mathcal{C}^i_j$ is to the left of interval $\mathcal{C}^i_l$. In Lemma 7, we establish properties of the cumulative search region. We show that every edge of the online matching M_i and the offline matching M^*_i is contained inside some interval of the sequence σ_i. We also show that there are no free servers inside any interval of σ_i, i.e., $S^i_F \cap \text{csr}_i = \emptyset$.

▶ **Lemma 7.** *After phase i of the algorithm,*

- *For every edge $(s, r) \in M^*_i \cup M_i$ there exists an interval $\mathcal{C} \in \sigma_i$ such that $(s, r) \subseteq \mathcal{C}$,*
- *The set of free servers S^i_F satisfies $S^i_F \cap csr_i = \emptyset$, and there is an interval $\mathcal{C}$ in σ_i such that the server $s_i \in bd(\mathcal{C})$.*

When a new request r_{i+1} is processed by the algorithm, the search interval $sr(r_{i+1})$ is added to the cumulative search region, i.e., $\text{csr}_{i+1} = \text{csr}_i \cup sr(r_{i+1})$. Suppose $\langle \mathcal{C}^i_j, \ldots, \mathcal{C}^i_l \rangle$ intersects $sr(r_{i+1})$, then csr_{i+1} will have a single interval that contains intervals $\langle \mathcal{C}^i_j, \ldots, \mathcal{C}^i_l \rangle$ and $sr(r_{i+1})$. From this description, an easy observation follows:

(O1) Given two intervals $\mathcal{C}$ and $\mathcal{C}'$ in cumulative search regions σ_i and σ_j respectively, with $j > i$, then either $\mathcal{C} \cap \mathcal{C}' = \emptyset$ or $\mathcal{C} \subseteq \mathcal{C}'$.

Matchings in cumulative search regions. From the property of cumulative search region established in Lemma 7, every edge of the online matching M_i and the offline matching M^*_i is contained inside some interval of the sequence σ_i. We denote the edges of online and offline matching that appear in an interval $\mathcal{C}$ of σ_i by $M_{\mathcal{C}}$ and $M^*_{\mathcal{C}}$ respectively. Therefore, $M_i = \bigcup_{\mathcal{C} \in \sigma_i} M_{\mathcal{C}}$ and $M^*_i = \bigcup_{\mathcal{C} \in \sigma_i} M^*_{\mathcal{C}}$. Note that $M_{\mathcal{C}}$ and $M^*_{\mathcal{C}}$ match the same set of servers and requests. Let this set of servers and requests be denoted by $S_{\mathcal{C}}$ and $R_{\mathcal{C}}$ respectively.

Consider any sequence of interior-disjoint intervals $\langle \mathcal{C}^{i_1}_{j_1}, \ldots, \mathcal{C}^{i_k}_{j_k} \rangle$, where each interval $\mathcal{C}^{i_l}_{j_l}$ appears in the cumulative search region σ_{i_l}. Note that every interval in this set can appear in after the execution of a different phase. In Lemma 8, we show that $\sum_{l=1}^{k} w(M^*_{\mathcal{C}^{i_l}_{j_l}}) \leq tw(M_{\text{OPT}})$, i.e., the total cost of the subset of offline matching edges that are contained inside all of these disjoint intervals is within a factor of t times the cost of the optimal matching. This property, therefore, relates the cost of subsets of offline matchings, each of which appear at different phases to the optimal cost.

▶ **Lemma 8.** *For any i, j, let $\mathcal{C}^i_j$ be the jth interval in the sequence σ_i. Suppose we are given intervals $\mathcal{C}^{i_1}_{j_1}, \mathcal{C}^{i_2}_{j_2} \ldots, \mathcal{C}^{i_k}_{j_k}$ such that no two of these intervals have any mutual intersection. Then $w(M^*_{\mathcal{C}^{i_1}_{j_1}}) + w(M^*_{\mathcal{C}^{i_2}_{j_2}}) + \ldots + w(M^*_{\mathcal{C}^{i_k}_{j_k}}) \leq tw(M_{\text{OPT}})$.*

4 Analysis for the line metric

For each interval of any cumulative search region σ_i, we assign it a level between 0 and $O(t \log(nt))$ based on the length of the interval. We also partition the edges of the online matching into $O(t \log(nt))$ levels.

Level of an interval. For any $0 < i \leq n$, we assign a *level* to every interval of any cumulative search region σ_i. The *level* of any interval $\mathcal{C}_j^i \in \sigma_i$, denoted by $lev(\mathcal{C}_j^i)$, is k if

$$(1 + \frac{1}{32t})^k (w(M_{\text{OPT}})/n) \leq \mathcal{L}(\mathcal{C}_j^i) \leq (1 + \frac{1}{32t})^{k+1} (w(M_{\text{OPT}})/n).$$

Here $\mathcal{L}(\mathcal{C}_j^i)$ is the length of the interval $\mathcal{C}_j^i$. All intervals whose length $\mathcal{L}(\mathcal{C}_j^i) \leq w(M_{\text{OPT}})/n$ is assigned a level 0.

Level of an online edge. For any request r_i and its match s_i in M_i, let r_i and s_i be contained in an interval $\mathcal{C}' \in \sigma_i$. Then, the level of this edge (r_i, s_i), denoted by $lev(r_i)$, is given by the level of the interval $\mathcal{C}'$, i.e., $lev(r_i) = lev(\mathcal{C}')$.

▶ **Lemma 9.** *The largest level of assigned to any online edge is* $O(t \log(nt))$.

Tracking the evolution of cumulative search region. The cumulative search region csr_i after any phase i will include disjoint intervals from the sequence σ_i. When we process a new request r_{i+1}, the cumulative search region is updated to include the open interval $sr(r_{i+1})$. This may combine a contiguous sub-sequence of intervals $\Gamma_{i+1} = \langle \mathcal{C}_j^i, \mathcal{C}_{j+1}^i, \ldots, \mathcal{C}_l^i \rangle$ of σ_i along with the interval $sr(r_{i+1})$ to form a single open interval $\mathcal{C}_j^{i+1}$ in σ_{i+1}. We refer to the sequence of intervals Γ_{i+1} as the *predecessors* of $\mathcal{C}_j^{i+1}$ and denote $i+1$ as the *birth phase* of $\mathcal{C}_j^{i+1}$. For each interval $\mathcal{C}$ in Γ_{i+1}, we define its *successor* to be $\mathcal{C}_j^{i+1}$ and denote $i+1$ as the *death phase* of $\mathcal{C}$. Suppose $\mathcal{C}_j^{i+1}$ is a level k interval. Then, it is easy to see that there is at most one level k interval in Γ_{i+1}.

▶ **Lemma 10.** *For* $k \geq 1$ *and any level* k *interval* $\mathcal{C}_j^{i+1}$*, there is at most one level* k *interval in the predecessor set* Γ_{i+1}.

Proof. For the sake of contradiction, suppose there are at least two interior-disjoint level k intervals $\mathcal{C}$ and $\mathcal{C}'$. Since $\mathcal{C}_j^{i+1}$ contains both $\mathcal{C}$ and $\mathcal{C}'$, its length is at least $2(1 + \frac{1}{32t})^k (w(M_{\text{OPT}})/n)$ contradicting the fact that $\mathcal{C}_j^{i+1}$ is a level k interval. ◀

Recollect that $M_{\mathcal{C}_j^{i+1}}$ and $M^*_{\mathcal{C}_j^{i+1}}$ are edges of the online and offline matching inside $\mathcal{C}_j^{i+1}$that match the servers of $S_{\mathcal{C}_j^{i+1}}$ to requests of $R_{\mathcal{C}_j^{i+1}}$. Let $M_{\Gamma_{i+1}} = \bigcup_{\mathcal{C} \in \Gamma_{i+1}} M_{\mathcal{C}}$ be the edges of the online matching contained inside the predecessors Γ_{i+1} of $\mathcal{C}_j^{i+1}$. By construction, the matchings $M_{\mathcal{C}_j^{i+1}}$ and $M_{\Gamma_{i+1}}$ only differ in the edge (s_{i+1}, r_{i+1}). So, $M_{\Gamma_{i+1}}$ match servers in $S_{\mathcal{C}_j^{i+1}} \setminus \{s_{i+1}\}$ to requests in $R_{\mathcal{C}_j^{i+1}} \setminus \{r_{i+1}\}$.

Maximal and minimal level k interval. A level k interval $\mathcal{C}$ is *maximal* if

- $\mathcal{C}$ is an interval in the cumulative search region after all the n requests have been processed, i.e., $\mathcal{C} \in \sigma_n$ or,
- if the successor $\mathcal{C}'$ of $\mathcal{C}$ has $lev(\mathcal{C}) > k$.

A level k interval $\mathcal{C}_j^{i+1}$ is a *minimal level k interval* if none of its predecessors in Γ_{i+1} are of level k.

Next, we will describe a sequence of nested level k intervals that capture the evolution of a minimal level k interval into a maximal level k interval. For any level k interval $\mathcal{C}$, we define a sequence of level k intervals $\mathcal{E}_{\mathcal{C}}$ along with a set $\text{comp}(\mathcal{C})$ of intervals of level strictly less than k in a recursive way as follows: Initially $\mathcal{E}_{\mathcal{C}} = \emptyset$ and $\text{comp}(\mathcal{C}) = \emptyset$. Suppose phase i is the birth phase of $\mathcal{C}$. If all the intervals in the predecessor Γ_i have a level less than

k, then $\mathcal{C}$ is a minimal level k interval. We add $\mathcal{C}$ to to front of the sequence $\mathcal{E}_{\mathcal{C}}$ and add the predecessors of $\mathcal{C}$, Γ_i, to the set comp($\mathcal{C}$) and return $\mathcal{E}_{\mathcal{C}}$ and comp($\mathcal{C}$). Otherwise, from Lemma 10, there is exactly one level k interval $\mathcal{C}' \in \Gamma_i$ and all other intervals have a level strictly less than k. In this case, we compute $\mathcal{E}_{\mathcal{C}'}$ and comp($\mathcal{C}'$) recursively. We set $\mathcal{E}_{\mathcal{C}}$ to the sequence $\mathcal{E}_{\mathcal{C}'}$ followed by $\mathcal{C}$. We add all intervals in $\Gamma_i \setminus \mathcal{C}'$ along with the intervals of comp($\mathcal{C}'$) to comp($\mathcal{C}$) and return comp($\mathcal{C}$) and $\mathcal{E}_{\mathcal{C}}$.

For any maximal level k interval $\mathcal{C}$, consider the sequence $\mathcal{E}_{\mathcal{C}} = \langle \mathcal{C}_1, \ldots, \mathcal{C}_l \rangle$ and the set of disjoint intervals comp($\mathcal{C}$) . From the construction of $\mathcal{E}_{\mathcal{C}}$, the following observations follow in a straight-forward way:

(E1) All intervals $\mathcal{C}' \in \mathcal{E}_{\mathcal{C}}$ are level k intervals. The first and the last interval, i.e., $\mathcal{C}_1$ and $\mathcal{C}_l = \mathcal{C}$ in $\mathcal{E}_{\mathcal{C}}$ are minimal and maximal level k intervals respectively,

(E2) The intervals in $\mathcal{E}_{\mathcal{C}}$ are nested, i.e., $\mathcal{C}_i \subseteq \mathcal{C}_{i+1}$ for all $1 \leq i \leq l-1$.

Let $\mathcal{M}_{\mathcal{C}}$ be the set of all level k edges of the online matching that are contained inside $\mathcal{C}$. Let $\mathcal{S}_{\mathcal{C}}$ and $\mathcal{R}_{\mathcal{C}}$ be the servers and request that are matched by $\mathcal{M}_{\mathcal{C}}$. From the construction of comp($\mathcal{C}$), the following properties follow in a straight-forward way:

(C1) comp($\mathcal{C}$) is a set of disjoint intervals each of which is contained inside $\mathcal{C}$,

(C2) Every point $s \in S_{\mathcal{C}} \setminus \mathcal{S}_{\mathcal{C}}$ and $r \in R_{\mathcal{C}} \setminus \mathcal{S}_{\mathcal{C}}$ is contained inside some interval from the set comp($\mathcal{C}$).

Let $I = \{I_1, \ldots, I_{l_k}\}$ be the set of all maximal level k intervals. In the following lemma, we show that I is a set of pairwise interior-disjoint intervals. Furthermore, the matchings $\mathcal{M}_{I_1}, \mathcal{M}_{I_2}, \ldots, \mathcal{M}_{I_{l_k}}$ partitions all the level k edges of the online matching M.

▶ **Lemma 11.** *Let $I = \{I_1, \ldots, I_{l_k}\}$ be the set of all maximal level k intervals. Then, I is a set of interior-disjoint intervals. Furthermore, the matchings $\mathcal{M}_{I_1}, \mathcal{M}_{I_2}, \ldots, \mathcal{M}_{I_{l_k}}$ partitions all the level k edges of the online matching M.*

For any level k interval $\mathcal{C}$, let $\mathcal{M}_{\mathcal{C}}^{\text{OPT}}$ be the optimal matching of $\mathcal{S}_{\mathcal{C}}$ and $\mathcal{R}_{\mathcal{C}}$. Suppose $\{I_1, \ldots, I_{l_k}\}$ be the set of all maximal level k intervals. Let $M^*_{\text{comp}(I_j)}$ be the union (over all intervals of comp(I_j)), the offline matching contained inside an interval of comp(I_j). By (C2) and Lemma 7, $M^*_{\text{comp}(I_j)}$ matches servers in $S_{I_j} \setminus \mathcal{S}_{I_j}$ to requests $R_{I_j} \setminus \mathcal{R}_{I_j}$. The symmetric difference of $\bigcup_{I_j \in I} M^*_{\text{comp}(I_j)}$and $\bigcup_{I_j \in I} M^*_{I_j}$consists of augmenting paths that match $\bigcup_{I_j \in I} \mathcal{S}_{I_j}$ to $\bigcup_{I_j \in I} \mathcal{R}_{I_j}$. Note that these are precisely the points that are matched by level k online edges. From Lemma 8, $w(\bigcup_{I_j \in I} M^*_{\text{comp}(I_j)}) \leq tw(M_{\text{OPT}})$and $w(\bigcup_{I_j \in I} M^*_{I_j}) \leq tw(M_{\text{OPT}})$ leading to the following lemma.

▶ **Lemma 12.** *For any $k > 0$, let $\{I_1 \ldots, I_{l_k}\}$ be the set of all maximal level k intervals. Then, $\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{OPT}}) \leq 2tw(M_{\text{OPT}})$.*

For $k > 0$, consider any maximal level k interval I_j. In Lemma 13, for $\varepsilon > 1/32t$ and $t = 3$, we show that the matching $\mathcal{M}_{I_j}$ is an ε-well-aligned matching of an ε-well-separated input instance. Additionally, we also show in this lemma that every far edge of this well-aligned matching is also a long edge of the online matching.

▶ **Lemma 13.** *For any level $k \geq 1$ interval $\mathcal{C}$, consider the sequence $\mathcal{E}_{\mathcal{C}}$. Let $\varepsilon = \frac{1}{32t}$ and $t = 3$. Then, $\mathcal{R}_{\mathcal{C}}$ and $\mathcal{S}_{\mathcal{C}}$ is an ε-well separated input and the matching $\mathcal{M}_{\mathcal{C}}$ is an ε-well-aligned matching of $\mathcal{S}_{\mathcal{C}}$ and $\mathcal{R}_{\mathcal{C}}$. Furthermore, each far edge in the ε-well aligned matching $\mathcal{M}_{\mathcal{C}}$ is also a long edge of the online matching.*

Let $\mathcal{M}_{I_j}^{\text{far}}$, $\mathcal{M}_{I_j}^{\text{close}}$ and $\mathcal{M}_{I_j}^{\text{med}}$ denote the far, close and medium edges of $\mathcal{M}_{I_j}$. Recollect that $\mathcal{M}_{I_j}^{\text{far}} \cup \mathcal{M}_{I_j}^{\text{close}} \cup \mathcal{M}_{I_j}^{\text{med}} = \mathcal{M}_{I_j}$. From Lemma 3, we know that

$$w(\mathcal{M}_{I_j}^{\text{close}}) + w(\mathcal{M}_{I_j}^{\text{med}}) \leq (2/\varepsilon + 3) w(\mathcal{M}_{I_j}^{\text{OPT}}) + \frac{4\varepsilon}{1-2\varepsilon} w(\mathcal{M}_{I_j}^{\text{far}}).$$

Adding the above inequality over all level k intervals and setting $\varepsilon = \frac{1}{32t}$ and $t = 3$, we get

$$\sum_{j=1}^{l_k}(w(\mathcal{M}_{I_j}^{\text{close}}) + w(\mathcal{M}_{I_j}^{\text{med}})) \leq (2/\varepsilon + 3)\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{OPT}}) + \frac{4\varepsilon}{1-2\varepsilon}\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{far}}).$$

Adding the above equation for all levels $k > 0$ and adding $w(M_{\text{OPT}})$ to the RHS and LHS, we get

$$w(M_{\text{OPT}}) + \sum_{k=1}^{O(\log n)}\sum_{j=1}^{l_k}(w(\mathcal{M}_{I_j}^{\text{close}}) + w(\mathcal{M}_{I_j}^{\text{med}})) \leq$$

$$w(M_{\text{OPT}}) + (\frac{2}{\varepsilon} + 3)\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{OPT}}) + \frac{2}{47}\sum_{k=1}^{O(\log n)}\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{far}}). \tag{9}$$

Note that every short edge of the online matching with level $k > 0$ will appear, for some maximal level k interval $I_j \in \{I_1, \ldots, I_{l_k}\}$, in $\mathcal{M}_{I_j}^{\text{close}}$ or $\mathcal{M}_{I_j}^{\text{med}}$. By construction, the short edges of level 0 have a length of at most $w(M_{\text{OPT}})/n$ and there are at most n such edges. Recollect that M_H is the set of short online edges. Therefore,

$$w(M_H) \leq w(M_{\text{OPT}}) + \sum_{k=1}^{O(\log n)}\sum_{j=1}^{l_k}(w(\mathcal{M}_{I_j}^{\text{close}}) + w(\mathcal{M}_{I_j}^{\text{med}})).$$

Every edge in $\mathcal{M}_{I_j}^{\text{far}}$ is only a long edge of the online matching. Recollect that M_L is the set of long edges of the online matching. We can, therefore, rewrite the (9) as

$$w(M_H) \leq O(1/\varepsilon)\sum_{k=1}^{O(\log n)}\sum_{j=1}^{l_k} w(\mathcal{M}_{I_j}^{\text{OPT}}) + \frac{2}{47}\, w(M_L),$$

$$w(M_H) \quad \leq \quad O(\log n)w(M_{\text{OPT}}) + \frac{2}{47}w(M_L).$$

The above inequality follows from Lemma 12 and the fact that every online edge appears in exactly one of the maximal level k intervals for some level k. In Lemma 2, by setting $t = 3$, we get $w(M_H) \geq w(M)/6$ and $w(M_L) \leq (5/6)w(M)$, and

$$\begin{aligned} w(M_H) - \frac{2}{47}w(M_L) &\leq O(\log n)w(M_{\text{OPT}}), \\ \frac{w(M)}{6} - \frac{2}{47}\times\frac{5}{6}w(M) &\leq O(\log n)w(M_{\text{OPT}}), \\ \frac{w(M)}{w(M_{\text{OPT}})} &\leq O(\log n). \end{aligned}$$

References

1 Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. A $o(n)$-competitive deterministic algorithm for online matching on a line. In *Approximation and Online Algorithms: 12th International Workshop, WAOA 2014*, pages 11–22, 2015.

2 Koutsoupias Elias and Nanavati Akash. The online matching problem on a line. In *Approximation and Online Algorithms: First International Workshop, WAOA 2003, Budapest, Hungary, September 16-18, 2003. Revised Papers*, pages 179–191, 2004.

3 A. Gupta and K. Lewi. The online metric matching problem for doubling metrics. In *Proceedings of International Conference on Automata, Languages and Programming*, volume 7391, pages 424–435, 2012.

4 Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993.

5 Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theor. Comput. Sci.*, 127(2):255–267, 1994.

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

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

8 Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 505–515, 2017.

9 Sharath Raghvendra. A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*, volume 60, pages 18:1–18:16, 2016.