SIGACT News Online Algorithms Column 28: Online Matching on the Line, Part 2

Rob van Stee
University of Leicester
United Kingdom

In the online matching problem on the line, requests (points in $\mathbb{R}$) arrive one by one to be served by a given set of servers. Each server can be used only once. This is a variant of the k-server problem restricted to the real line. Although easy to state, this problem is stil wide open. The best known lower bound is 9.001 [2], showing that this problem is really different from the well-known cow path problem. Antoniadis et al. [1] recently presented a sublinearly competitive algorithm.

In this column, I present some results by Elias Koutsoupias and Akash Nanavati on this problem with kind permission of the authors. The column is based on Akash' PhD thesis [4], which contains an extended version of their joint WAOA 2003 paper [3] which has never appeared in a journal. I have expanded the proofs and slightly reorganized the presentation.

The previous column (see SIGACT News 47(1):99-111) contains a proof of a linear upper bound for the generalized work function algorithm and a logarithmic lower bound for the algorithm. This column gives a more detailed analysis of this algorithm, leading to a different (but again linear) upper bound. The techniques used here may potentially be helpful to show a sublinear upper bound for γ-WFA. I conjecture that this algorithm in fact has a logarithmic competitive ratio (which would match the known lower bound for it), but this very much remains an open question.

1 Introduction

We begin by repeating some definitions for convenience.

In this column, both requests and servers are specified by points on the real line and are multisets, as there can be multiple requests and/or servers at the same location. The same holds for all other sets discussed below. For example, for $x \in I$, we have $I \subsetneq I \cup \{x\}$. We will use the notation $\{x\}^k$ to denote a set which contains k copies of the point x. Time is discrete, with one request occurring per time step.

Let R^t denote the set of requests until time t and let A^t denote the set of servers used by the online algorithm to match these requests. Of course $|A^t| = |R^t|$. Let $M(A^t, R^t)$ denote a way of matching R^t to A^t that minimizes the total offline cost. This matching can be obtained by matching the requests to servers in order from left to right. We denote its cost by $\textsc{pseudo}_t$, whereas the *optimal* cost of serving the first t requests is denoted by $\textsc{opt}_t$. Clearly, $\textsc{opt}_t \leq \textsc{pseudo}_t$ for $t = 1, \ldots, n$, and $\textsc{opt}_n = \textsc{pseudo}_n$.

Given this definition of the matching $M(A^t, R^t)$ we can ask how many of its lines cross a point $x \in \mathbb{R}^t$. We denote this number by $\textsc{cross}_x(A^t, R^t)$ or simply $\textsc{cross}_x$. To define it properly let $\textsc{left}_x(I)$ denote the number of elements of the set I to the left of x. Then $\textsc{cross}_x(A^t, R^t) = \textsc{left}_x(A^t) - \textsc{left}_x(R^t)$. We have the following property:

$$\textsc{pseudo}_t = \int_{-\infty}^{\infty} |\textsc{cross}_x(A^t, R^t)| dx \qquad (1)$$

As stated in the previous column (see Section 4 of that column), for all $i = 1, \ldots, n$, we have the following bound for the cost of γ-WFA to serve the ith request:

$$\begin{aligned} d(r_i, s_i) &\leq \textsc{pseudo}_{i-1} + \textsc{pseudo}_i \\ &\leq \frac{\gamma+1}{\gamma-1}(\textsc{opt}_{i-1} + \textsc{opt}_i) \qquad \text{using Theorem 4} \\ &\leq \frac{2(\gamma+1)}{\gamma-1}\textsc{opt}_i \end{aligned}$$

Summing this over all i gives

$$\gamma\text{-WFA}(\sigma) \leq \frac{2(\gamma+1)}{\gamma-1} \sum_{i=1}^{n} \textsc{opt}_i \leq \frac{2n(\gamma+1)}{\gamma-1}\textsc{opt}(\sigma),$$

which is $O(n) \cdot \textsc{opt}(\sigma)$ for all $\gamma > 1$.

In Section 3, we give a bound in terms of the number of crossing lines in the matching produced by γ-WFA. In Section 4, we consider the concept of extended costs. In Section 5, we define multipliers and use them to derive a different (but still linear) upper bound on the competitive ratio of γ-WFA.

2 Segments

A *free interval* is a maximal interval on the line that does not contain any requests or servers. When γ-WFA matches a request r to some server, we will denote this server by s_r and the interval (or edge) (r, s_r) or (s_r, r) by E_r. This interval is partitioned into a set of segments, one for each

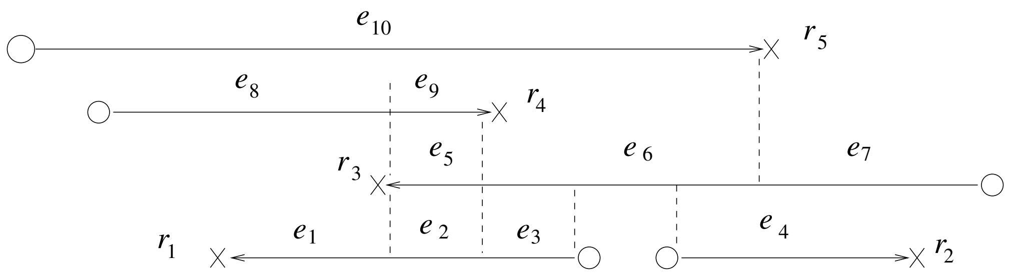

Figure 1: Types of segments

free interval. This set is also denoted by E_r. The general idea of the proof is to associate with each edge E_r a set of segments of older requests which are *cancelled* by the edge.

We say that r is a *left* (*right*) request if it is serviced by a server on its left (right). This orientation extends naturally to the segments of E_r: if r is left, then so is every segment of E_r. By symmetry, every property that we show below for left requests has a counterpart for right requests. When we have a choice, we consider left requests.

We refer to the state immediately after the t-th request is matched as time t. Let $E^t_r(I)$ be the set of edges created by γ-WFA after r and at or before time t which include the segment I.

Definition 4 *A segment $I \in E_r$ is* alive *at time r' if for every $t = r+1, \dots, r'$, at least half of the edges in $E^t_r(I)$ has the same orientation as I.*

Definition 5 *An alive segment I is* exposed *at time r' if exactly half the edges in $E^{r'}_r(I)$ are left edges. An alive segment which is not exposed is called* protected.

Definition 6 *A segment is* cancelled *by request r' if it ceases to be alive at time r'.*

This means that after r' was serviced, the number of edges with the opposite orientation to r in $E^{r'}_r(I)$ is larger than the number of edges with the same orientation for the first time. (Just before r' arrived, I was exposed.) In particular, we have the following important property.

Property 2 *A request r' which cancels a segment I of request r has the opposite orientation to r (and hence to I).*

In Figure 1, circles represent servers and crosses represent requests. Higher requests appeared later, so r_3 came after r_1 and r_2. The requests r_1 and r_2 are right requests, the others are left requests. The edge E_{r_1} is partitioned into the segments e_1, e_2, e_3. This partitioning happens gradually as later requests arrive and are served. Request r_3 kills $E_{r_2} = \{e_4\}$ and protects the segments e_2 and e_3. After request r_4 is served, the segment e_1 is cancelled, and e_2 is exposed. Note that at this time, the segments e_2 and e_9, which are in opposite directions but cover the same interval, are both alive (whereas e_5 has been cancelled by e_9).

Lemma 4 (Prefix Aliveness) *Let r be a left request matched to s_r. Consider some later time r'. If some part of E_r is alive at time r', this part is a contiguous interval (s_r, t) for some $s_r < t \le r$. If $t < r$, t is the leftmost right request occuring after time r and before or at time r'.*

Proof Whether any segment of E_r is still alive at time r' is determined by the relative number of right and left requests that arrive after r and that contain this segment. If there are no right requests which contain any segment in (s_r, r), then (s_r, r) is alive in its entirety and the lemma holds.

Otherwise, by locality, γ-WFA uses only surrounding servers for any request (see Property 1 in the previous column). This means that after r is served, there are no unmatched servers in the interval (s_r, r). Any right request (see Property 2) that cancels a segment of E_r, but does not cancel E_r completely, is in (s_r, r) and must be matched to a server to the right of r. It follows by induction that each time that this happens, a (possibly empty) *suffix* of the alive part of E_r is cancelled, and each time the leftmost point of a newly cancelled part is the leftmost right request that has occurred so far inside (s_r, r). Hence the alive part of E_r is always a prefix of (s_r, r), and must be contiguous. $\square$

Denote the alive part of E_r after some later request r' has been served by $\text{ALIVE}(r, r')$.

Lemma 5 (Contiguity of Exposed) *Let r be a left request matched to s_r. Consider some later time r'. At this time, the set of exposed segments of E_r is contiguous, and bounded from the left by the rightmost left request in (s_r, r) that arrived so far. There is no request inside this set of exposed segments at any time between r and r'.*

Proof Any exposed segment of E_r is alive. If there is no exposed segment, there is nothing to show. If there is a part I of $\text{ALIVE}(r, r')$ that is protected at time r', there must be a left request that arrives after r and that contains I. By locality (Property 1 in the previous column), γ-WFA uses only surrounding servers for matching. Hence, there is no unused server in the interval (s_r, r), and such a left request must be matched to a server to the left of s_r. This implies immediately that at any time at which some segment of E_r is alive and exposed, a (possibly empty) prefix of E_r is protected, and that it is protected from s_r up to the rightmost left request in (s_r, r) that arrived so far. Hence both the protected and exposed parts of $\text{ALIVE}(r, r')$ are contiguous.

The final statement in the lemma follows immediately from the bounds on the exposed set of segments in this lemma and in Lemma 4. $\square$

In fact it can be seen from these proofs that the level of protection is monotonically decreasing from left to right in (s_r, r) for any left request r of which some part is still alive. Here the level of protection at a point x is defined as the number of left edges arriving after r and crossing x minus the number of right edges arriving after r and crossing x. (The part of E_r that is cancelled is thus "negatively protected", with the protection level still monotonically decreasing from left to right.)

3 Unerased vs. Pseudo-Optimal

We say that a point x contributes to the value PSEUDO_i at time $t \leq i$ if we have $|\text{CROSS}_x(A_t, R_t)| > |\text{CROSS}_x(A_{t-1}, R_{t-1})|$ and the absolute value never drops below $|\text{CROSS}_x(A_t, R_t)|$ afterward. More generally, x contributes to UNERASED_i at time $t \leq i$ if $\text{CROSS}_x(A_t, R_t) \neq \text{CROSS}_x(A_{t-1}, R_{t-1})$, and we also have $\text{CROSS}_x(A_{t'}, R_{t'}) \neq \text{CROSS}_x(A_{t-1}, R_{t-1})$ for all $t < t' \leq i$. That is, after time $t-1$, the number of lines crossing x was never again equal to $\text{CROSS}_x(A_{t-1}, R_{t-1})$. (Note that this integer value changes by at most one each time that a request is served.)

UNERASED_i counts for each point all edges that cross this point and that are not cancelled out by later requests (up to and including at time i). The number of times that a point x contributes to $\textsc{unerased}_i$ is

$$\max_{1\le t\le i-1} \textsc{cross}_x(A_t, R_t) - \min_{1\le t\le i-1} \textsc{cross}_x(A_t, R_t)$$

since for each value in this range, x contributes (only) at the last time at which this value is reached. We say that a point contributes k times to $\textsc{unerased}_i$ if it contributes to $\textsc{unerased}_i$ at exactly k distinct times.

Note that the number of times that any given point x contributes to $\textsc{unerased}_i$ is monotonically nondecreasing from time 1 to time i: if a segment containing x is created at time i_1 and canceled at time i_2, then it does not contribute at time i_1, but it does at time i_2. Thus, the contribution at time i_1 is not lost but replaced by a new contribution (by a request in the other direction).

This also shows that $\textsc{unerased}_i$ itself is monotonically nondecreasing in i: the number of times any given point contributes can only go up over time.

Theorem 9

$$\frac{\gamma-1}{2\gamma}\textsc{unerased}_i \le \textsc{pseudo}_i \le \textsc{unerased}_i \qquad i = 1, \ldots, n.$$

Proof By (1), $\textsc{pseudo}_i$ is the measure of all points, each point taken with the multiplicity equal to the number of times that it contributes to $\textsc{pseudo}_i$. Furthermore $\textsc{unerased}_i$ is the measure of all points, each point taken with the multiplicity equal to the number of times that it contributes to $\textsc{unerased}_i$. Since the condition for this is weaker than for contributing to $\textsc{pseudo}_i$, the second inequality follows.

Consider a request r_t which is serviced by request s and assume without loss of generality that $s < r_t$. By Lemma 4, the points that contribute to $\textsc{unerased}_i$ at time t form an interval (s, q) for some point $q \in [s, r_t]$. (If E_{r_t} gets cancelled by time i, then $q = s$ and the interval of points contributing at time t is empty.)

Lemma 2 from the previous column implies directly that at least a fraction $\frac{\gamma-1}{2\gamma}$ of this interval has $\textsc{cross}_x(A_{t-1}, R_{t-1}) \ge 0$. This fraction contributes also to $\textsc{pseudo}_i$ at time t, which proves the lower bound on $\textsc{pseudo}_i$. □

Again using Theorem 4, we conclude that

$$\textsc{opt}_i \le \textsc{pseudo}_i \le \textsc{unerased}_i \le \frac{2\gamma}{\gamma-1}\textsc{pseudo}_i \le \frac{2\gamma}{\gamma-1}\frac{\gamma+1}{\gamma-1}\textsc{opt}_i.$$

Furthermore, the cost to service request r_i is at most equal to $\textsc{unerased}_i$, since each point in the interval (r_i, s) definitely contributes to $\textsc{unerased}_i$ (the edge E_r is new and therefore not cancelled). Since as noted above, $\textsc{unerased}_i$ is monotonically nondecreasing, we can bound the total cost by

$$\gamma\text{-}\textsc{wfa} \le \sum_{i=1}^{n} \textsc{unerased}_i \le n \cdot \textsc{unerased}_n \le \frac{2\gamma}{\gamma-1}\frac{\gamma+1}{\gamma-1} n \cdot \textsc{opt}_n.$$

Theorem 10 γ-$\textsc{wfa}$ *has a competitive ratio of at most* $\frac{2\gamma}{\gamma-1}\frac{\gamma+1}{\gamma-1} \cdot n = O(n)$*, where* n *is the number of requests.*

4 Forest

We construct a forest where nodes are requests. Higher nodes in a tree (closer to the root) represent later requests. The forest is inductively defined for each balanced interval. Here a balanced interval may be equal to the entire real line or a halfline, in order to cover the very last request in the input and any other requests that occur to one side of all the remaining servers. (In this case, there is not really a server at one or both ends of the interval.)

The forest for a balanced interval (s, s') is created as follows. Let r be the last request in the interval and let the server it is matched to be s_r. Then WLOG $s < s_r < r < s'$. We let r be the root of the tree. Its children are the roots of the trees defined recursively in the forest in (s_r, s'). The requests in (s, s_r) form a separate forest. So any earlier request which was to the right of s_r is a child of r, whereas any earlier request to the left of s_r is not.

The children of r are ordered from left to right according to their relative positions on the line. That is, if $r_1, \dots, r_k$ are the roots of trees in the forest of (s_r, s'), such that $s < r_1 < \cdots < r_k < s'$, then they are children of r in that left to right order.

Let us label nodes of this forest by the type of the corresponding request, i.e., L for left and R for right requests. Observe that labels of nodes when put in sequence form words from the regular expression R^*L^*. Also, if r_j is the rightmost right request and r_{j+1} is the leftmost left request in (s, s'), then the order of arrival of the requests is $r_1 > \cdots > r_j$ and $r_{j+1} < \cdots < r_k$. There is no particular ordering of the left requests relative to the right requests.

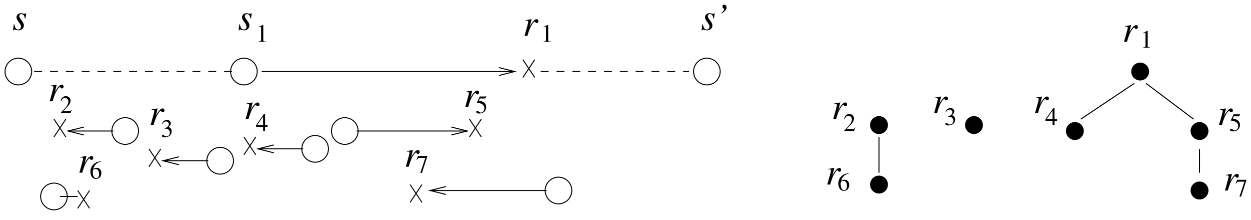

Figure 2: An example forest. Request r_i is served by server s_i $(i = 1, \dots, 7)$. The roots are r_2, r_3, r_1, forming the word RRL. The children of r_1 form the word RL. Note that r_3 arrived before r_2; if it had not, s_2 would have already been used when r_3 arrived, and in our construction r_3 would have become a root of a tree in the interval (s, s_3), with r_2 as its child.

Lemma 6 (Advantage of WFA) *Let (s, s') be a balanced interval. Let A, R denote the servers and requests inside this interval, so that γ-WFA matched R to A. Let $r < r'$ denote two requests in this interval that arrived in the order r, r' so that γ-WFA matched r to s' and r' to s. Then*

$$3(M(A + s + s', R + r + r') - M(A + s', R + r')) + d(s, r') \geq 2d(r, r').$$

Proof Let $B_1^j = ||B^j(A, R) \cap (s, r)||, B_2^j = ||B^j(A, R) \cap (r, r')||, B_3^j = ||B^j(A, R) \cap (r', s')||$. Define B_1^+, B_1^0, B_1^- etc. appropriately. Let $B_1 = B_i^+ + B_i^0 - B_i^-$ and $B_i = B_i^- + B_i^0 - B_i^+$ for $i = 2, 3$. (Note the difference, caused by the directions in which the various servers travel in the

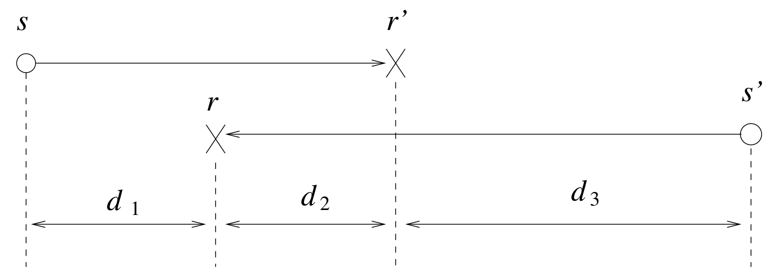

Figure 3: The situation considered in Lemma 6.

scenarios considered below.) Then

$$\begin{aligned} M(A+s,R+r) &= M(A,R)+B_1 \\ M(A+s',R+r) &= M(A,R)+B_2+B_3 \\ M(A+s+s',R+r+r') &= M(A,R)+B_1+B_3. \end{aligned}$$

When 3-WFA matches r to s', the following inequality holds:

$$\begin{aligned} & 3M(A+s,R+r)+d(s,r) \geq 3M(A+s',R+r)+d(r,s') \\ & \qquad \Leftrightarrow 3B_1+d_1 \geq 3B_2+3B_3+d_2+d_3. \end{aligned}$$

For the quantity that we wish to bound from below, we therefore find

$$\begin{aligned} 3(M(A+s+s',R+r+r')-M(A+s',R+r))+d(s,r') &= 3(B_1-B_2)+d_1+d_2 \\ &\geq 2d_2+3B_3+d_3. \end{aligned}$$

Moreover, $3B_3+d_3 = 3(B_3^- + B_3^0 - B_3^+) + B_3^+ + B_3^0 + B_3^- = 4(B_3^- + B_3^0) - 2B_3^+$, which is nonnegative by Lemma 2 of the previous column. The lemma follows. □

This lemma shows that the cancelled part, $d(r,r')$, is not larger than the increase in PSEUDO. We are now ready to prove the most crucial lemma in our proof.

Definition 7 *Let ℓ be a request that arrived in a balanced interval (s,s') with servers A and requests R. Then* LEFT$(\ell) = M(A+s,R+\ell)$ *and* RIGHT$(\ell) = M(A+s',R+\ell)$.

Lemma 7 (Segment cancelling) *Let ℓ be a left node and let v be its right descendant. Assume the subinterval of $E_\ell \cap E_v$ that is cancelled by ℓ is nonempty and denote it by (e,f). Then*

$$3(\text{LEFT}(\ell) - \sum_{c \in CH(\ell)} \text{LEFT}(c)) + 3d(s_\ell,\ell) \geq 4d(e,f)$$

where $CH(\ell)$ denotes the set of children of ℓ.

Proof Let $CH(\ell) = \{r_1,\ldots,r_i,\ell_k,\ldots,\ell_1\}$. Suppose v is in the subtree T_d, where r_d labeled R is a child of ℓ. The case when v is in a subtree of a left child is analogous. We have $M(A_\ell+s_\ell,R_\ell+\ell) \geq M(A_\ell+s_\ell,R_\ell+f)-d(f,\ell)$, since in the best (cheapest) case for $M(A_\ell+s_\ell,R_\ell+\ell)$, the interval (f,ℓ) does not need to be covered anymore after ℓ arrives, whereas it was covered before.

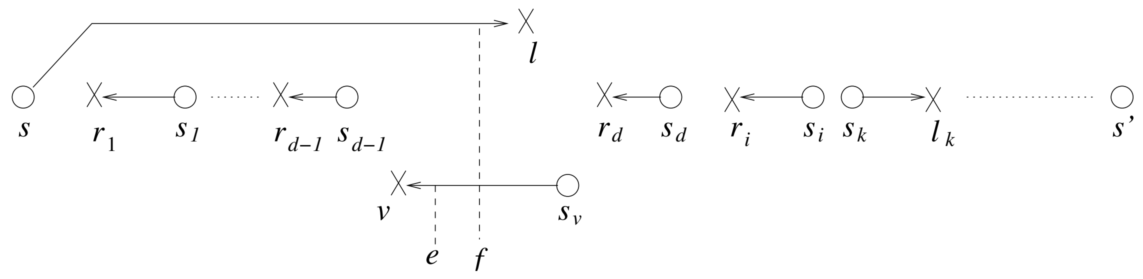

Figure 4: The situation considered in Lemma 7. Note that the interval (v, e) has been cancelled by other requests (descendants of r_d) that are not shown in this figure, whereas the interval (f, ℓ) does not get cancelled because it was protected by left requests that are not shown. Of course, $v = e$ and $f = \ell$ (and/or $f = s_v$) are possible, as well as $v = r_d$ (instead of a descendant of r_d).

In order to bound $M(A_\ell+s_\ell, R_\ell+f)$, we use that by our construction of the forest for ℓ, we have $r_1 > \cdots > r_i$. Thus r_1 arrived last and the choices of servers for 3-WFA were s and s_1; generally, the choices of servers to serve r_j were s_{j-1} and s_j for $j = 1, \ldots, i$ (i.e., for all the right children of ℓ). We can make a similar observation for the left children. This means that $M(A_\ell + s_\ell, R_\ell + f)$ can be expressed in terms of RIGHT(c) for all children of ℓ up to r_{d-1}, and in terms of LEFT(c) for all children starting from r_{d+1}, since none of these changed after such a child was served. We get

$$\begin{aligned}&\textsc{left}(\ell) + d(f,\ell) \ge M(A_\ell + s_\ell, R_\ell + f)\\ &= \sum_{j=1}^{d-1}\textsc{right}(r_j) + \sum_{j=d+1}^{i}\textsc{left}(r_j) + \sum_{j=1}^{k}\textsc{left}(\ell_j) + M(A_{r_d} + s_{d-1} + s_d, R_{r_d} + r_d + f).\end{aligned} \tag{2}$$

We also have

$$d(s,\ell) = \sum_{j=1}^{d-1} d(s_{j-1}, s_j) + d(s_{d-1}, f) + d(f,\ell) \tag{3}$$

(where we define $s_0 = s$) and

$$\textsc{right}(r_j) \ge \textsc{left}(r_j) - d(s_{j-1}, s_j) \qquad j = 1, \ldots, d-1. \tag{4}$$

(If $\textsc{right}(r_j) < \textsc{left}(r_j) - d(s_{j-1}, s_j)$, then 3-WFA would have served r_j from the right, since $3\textsc{right}(r_j) + d(s_{j-1}, r_j) < 3\textsc{left}(r_j) - 2d(s_{j-1}, s_j) < 3\textsc{left}(r_j) + d(s_j, r_j)$.) By (2)–(4),

$$\begin{aligned}\textsc{left}(\ell) \ge &\sum_{j=1}^{d-1}(\textsc{left}(r_j) - d(s_{j-1}, s_j)) + \sum_{j=d+1}^{i}\textsc{left}(r_j) + \sum_{j=1}^{k}\textsc{left}(\ell_j)\\ &+ M(A_{r_d} + s_{d-1} + s_d, R_{r_d} + r_d + f) - d(f,\ell)\\ = &\sum_{c\in CH(\ell)}\textsc{left}(c) - \textsc{left}(r_d) - (d(s,\ell) - d(s_{d-1}, f)) + M(A_{r_d} + s_{d-1} + s_d, R_{r_d} + r_d + f).\end{aligned} \tag{5}$$

We will express $\textsc{left}(r_d)$ and $M(A_{r_d} + s_{d-1} + s_d, R_{r_d} + r_d + e)$ in terms of $\textsc{left}(v)$ and $M(A_v + s'_v + s_v, R_v + v + e)$ in order to apply Lemma 7. By Lemma 5, there is no request after v in the

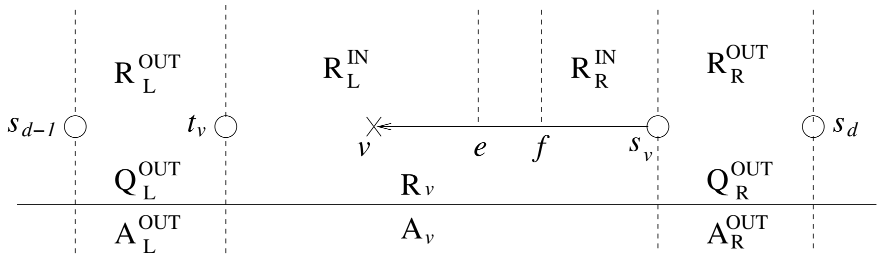

Figure 5: Zooming in on request v and what happened afterwards. Compare Figure 4.

interval (e, f). We partition the set of requests that arrived after v and before ℓ as follows. Let t_v be the closest server to the right of v when v arrived (the one that 3-WFA also considered using).

- requests $R_{\mathrm{R}}^{\mathrm{OUT}}$ in the interval (s_{d-1}, t_v)
- requests $R_{\mathrm{L}}^{\mathrm{IN}}$ in the interval (t_v, e)
- requests $R_{\mathrm{R}}^{\mathrm{IN}}$ in the interval (f, s_v)
- requests $R_{\mathrm{R}}^{\mathrm{OUT}}$ in the interval (s_v, s_d).

We also partition requests that arrived before v along with the servers used to serve them:

- requests $Q_{\mathrm{L}}^{\mathrm{OUT}}$, servers $A_{\mathrm{L}}^{\mathrm{OUT}}$ in the interval (s_{d-1}, t_v)
- requests $Q_{\mathrm{R}}^{\mathrm{OUT}}$, servers $A_{\mathrm{R}}^{\mathrm{OUT}}$ in the interval (s_v, s_d).

The requests in (t_v, s_v) before v are denoted by A_v as usual; they were served by servers R_v in (t_v, s_v). See Figure 5.

Request r_d arrived after v and is to the right of f, since (e, f) was exposed when ℓ arrived. In the proof below, we consider the case that $r_d \in R_{\mathrm{RIGHT}}^{\mathrm{IN}}$. The case $r_d \in R_{\mathrm{RIGHT}}^{\mathrm{OUT}}$ is similar.

We have $|A_v| = |R_v|$, and (e, f) was exposed after r_d arrived and until ℓ arrived, so when ℓ arrives no line in the matching of 3-WFA crosses the interval (e, f) or the points s_{d-1} and s_d. Therefore $|Q_{\mathrm{L}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{OUT}} + R_{\mathrm{L}}^{\mathrm{IN}}| = |A_{\mathrm{L}}^{\mathrm{OUT}} + t_v|$ and $|Q_{\mathrm{R}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{IN}}| = |A_2 + s_d|$. Hence, in the pseudo-optimal matching LEFT(r_d), $x(t) = |R_{\mathrm{L}}^{\mathrm{IN}}|$ lines cross (or reach!) t_v and $x(v) = |R_{\mathrm{R}}^{\mathrm{IN}}|$ lines cross s_v. By cutting these lines at t_v and s_v, we can write its cost as follows.

$$\begin{aligned}\text{LEFT}(r_d) &= M(A_{r_d} + s_d, R_{r_d} + r_d)\\ &= M(A_{\mathrm{L}}^{\mathrm{OUT}} + t_v, Q_{\mathrm{L}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{OUT}} + t_v^{x(t)}) + M(A_{\mathrm{R}}^{\mathrm{OUT}} + s_d, Q_{\mathrm{R}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{OUT}} + s_v^{x(v)})\\ &\quad + M(A_v + t_v^{x(t)} + s_v + s_v^{x(t)}, R_v + R_{\mathrm{L}}^{\mathrm{IN}} + v + R_{\mathrm{R}}^{\mathrm{IN}}).\end{aligned}$$

Using similar reasoning (but this time assigning t_v to the middle part), we can write

$$\begin{aligned}M(A_{r_d} + s_d + s_{d-1}, R_{r_d} + r_d + f) &= M(A_{\mathrm{L}}^{\mathrm{OUT}} + s_{d-1}, Q_{\mathrm{L}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{OUT}} + t_v^{x(t)})\\ &\quad + M(A_{\mathrm{R}}^{\mathrm{OUT}} + s_d, Q_{\mathrm{R}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{OUT}} + s_v^{x(v)})\\ &\quad + M(A_v + t_v^{x(t)} + t_v + s_v + s_v^{x(v)}, R_v + R_{\mathrm{R}}^{\mathrm{IN}} + v + f + R_{\mathrm{R}}^{\mathrm{IN}})\end{aligned}$$

Similarly to (4), we have $M(A_{\mathrm{L}}^{\mathrm{OUT}} + s_{d-1}, Q_{\mathrm{L}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{OUT}} + t_v^{x(t)}) \geq M(A_{\mathrm{L}}^{\mathrm{OUT}} + t_v, Q_{\mathrm{L}}^{\mathrm{OUT}} + R_{\mathrm{R}}^{\mathrm{OUT}} + t_v^{x(t)}) - d(s_{d-1}, t_v)$. Starting from (5), we conclude

$$\begin{aligned}
&3(\textsc{left}(\ell) - \sum_{c \in CH(\ell)} \textsc{left}(c)) + 3d(s_\ell, \ell) \\
&\geq 3(M(A_{r_d} + s_{d-1} + s_d, R_{r_d} + r_d + f) - \textsc{left}(r_d)) + 3d(s_{d-1}, f) \\
&\geq 3M(A_v + t_v^{x(t)} + t_v + s_v + s_v^{x(v)}, R_v + R_{\mathrm{R}}^{\mathrm{IN}} + v + f + R_{\mathrm{R}}^{\mathrm{IN}}) \\
&\ - 3M(A_v + t_v^{x(t)} + s_v + s_v^{x(t)}, R_v + R_{\mathrm{L}}^{\mathrm{IN}} + v + R_{\mathrm{R}}^{\mathrm{IN}})) + 3d(t_v, f) \\
&\geq 3(M(A_v + t_v + s_v, R_v + v + f) - M(A_v + s_v, R_v + v)) + 3d(t_v, f) \\
&\geq 2d(v, f) + 2d(t_v, f) \\
&= 4d(v, f) + 2d(t_v, v) \\
&\geq 4d(v, f) \\
&\geq 4d(e, f).
\end{aligned}$$

Here we have applied the quasi-convexity theorem (Theorem 2) for the third inequality, and Lemma 6 for the fourth inequality. □

5 Multipliers

Let $I_1, \ldots, I_k$ be the set of maximal contiguous segments cancelled by ℓ. Suppose inductively that we have multipliers m_i for each request corresponding to the segments $I_1, \ldots, I_k$. Let $j = \arg\max m_i$. Define

$$m_\ell = 1 + \max(m_j - 1, \max_{i \neq j} m_i).$$

That is, m_ℓ is the maximum of the multipliers of its cancelled descendants unless two of them achieve the maximum value, in which case m_ℓ is raised by 1. If there are no segments cancelled by ℓ, define $m_\ell = 1$.

Property 3 *For $i \neq j = \arg\max m_i$, we have $m_\ell \geq m_i + 1$ and $m_\ell \geq m_v$. Furthermore $m_\ell \leq m_j + 1$, and $m_\ell = m_j + 1 \Leftrightarrow m_j = m_i$ for some $i \neq j$.*

For the following lemma, recall that $\textsc{alive}(a, b)$ is the part of E_a that is still alive after request b has been served.

Lemma 8 (Total benefit of cancelling) *Let T_ℓ denote the set of nodes in the subtree rooted at request ℓ. Then*

$$3\textsc{left}(\ell) + 4 \sum_{i \in T_\ell} m_i ||\textsc{alive}(i, \ell)|| \geq \sum_{i \in T_\ell} d(i, s_i).$$

Proof We use induction. The base case is trivial: if a tree consists of a single node ℓ, then E_ℓ is completely alive, $m_\ell = 1$ and $||\textsc{alive}(\ell, \ell)|| = d(\ell, s_\ell)$.

For the induction step, we need to prove the following. Recall that $\{I_i\}_{i=1}^k$ is the set of segments cancelled by ℓ.

$$3(\textsc{left}(\ell) - \sum_{c \in CH(\ell)} \textsc{left}(c)) + 4m_\ell ||\textsc{alive}(\ell, \ell)|| \geq d(\ell, s_\ell) + 4 \sum_{i=1}^{k} m_i ||I_i||.$$

Clearly, $||\textsc{alive}(\ell,\ell)|| = d(\ell,s_\ell) > \sum_{i=1}^{k} ||I_i||$. By Property 3, we have

$$4(m_\ell - 1)||\textsc{alive}(\ell,\ell)|| \geq 4\sum_{i\neq j} m_i||I_i|| + 4(m_j-1)||I_j||.$$

This means that we need to show

$$3(\textsc{left}(\ell) - \sum_{c\in CH(\ell)} \textsc{left}(c)) + 4||\textsc{alive}(\ell,\ell)|| \geq d(\ell,s_\ell) + 4||I_j||. \tag{6}$$

We have $||\textsc{alive}(\ell,\ell)|| = d(\ell,s_\ell)$, so (6) follows from Lemma 7. □

Unfortunately, the only bound on m_i that we have is that $m_i \leq n$ for $i = 1,\ldots,n$. Thus all we can conclude from this lemma is that the total online cost is upper bounded by

$$3\textsc{pseudo}_n + 4n\sum_{i\in T_n} ||\textsc{alive}(i,\ell)||.$$

Now $\sum_{i\in T_n} ||\textsc{alive}(i,\ell)||$ is exactly $\textsc{unerased}_n$. For $\gamma = 3$, we have $\textsc{unerased}_n \leq 3\textsc{pseudo}_n$ by Theorem 9, and $\textsc{pseudo}_n \leq 2\textsc{opt}_n$. We conclude that the total online cost is at most $(12n + 3)\textsc{pseudo}_n \leq (24n+6)\textsc{opt}_n$.

In some cases, for instance the lower bound shown in Theorem 8 in the previous column, the multipliers do not get higher than $\log n$ and in this case, a logarithmic upper bound follows immediately. This happens on any input for which requests are never protected or partially cancelled. For general inputs, it appears the maximum multiplier can be $\Omega(n)$.

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 Evripidis Bampis and Ola Svensson, editors, *Approximation and Online Algorithms - 12th International Workshop, WAOA 2014, Wrocław, Poland, September 11-12, 2014, Revised Selected Papers*, volume 8952 of *Lecture Notes in Computer Science*, pages 11–22. Springer, 2014.

[2] Bernhard Fuchs, Winfried Hochstättler, and Walter Kern. Online matching on a line. *Theor. Comput. Sci.*, 332(1-3):251–264, 2005.

[3] Elias Koutsoupias and Akash Nanavati. The online matching problem on a line. In Klaus Jansen and Roberto Solis-Oba, editors, *Approximation and Online Algorithms, First International Workshop, WAOA 2003, Budapest, Hungary, September 16-18, 2003, Revised Papers*, volume 2909 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2003.

[4] Akash Nanavati. *Coordination Mechanisms and Online Matching.* PhD thesis, University of California, Los Angeles, 2004.