Lowerbounds for the Online Minimum Matching Problem on the Line

Maximillian C. W. Bender
Department of Computer Science
Connecticut College
New London, Connecticut 06320
Email: mbender1@conncoll.edu

Advisor - Christine Chung
Department of Computer Science
Connecticut College
New London, Connecticut 06320
Email: cchung@conncoll.edu

***Abstract*—We present lowerbounds for the competitive ratio of randomized algorithms on the *online minimum matching problem* on the line for both the *minimum weight* and *bottleneck* objectives. To the best of our knowledge, there is no established lowerbound for randomized algorithms for either objective on the line: we show a lowerbound of 2 for the minimum weight objective and a linear lowerbound for the bottleneck objective, hence proving that the bottleneck objective is a harder problem than the minimum weight objective on the line. We also present a tool for establishing a higher lower bound for the minimum weight objective.**

I. Introduction

We consider the online minimal metric matching problem, where we are a priori given a set of servers in a metric space in which a request sequence of equal length will arrive in locations unknown until arrival. As each request arrives, it must be irrevocably matched to a server. Each server can only be matched to one request. The two different goals we discuss for the problem are 1) the *minimum weight* objective, which minimizes the average (or equivalently total) distance between any request and its paired server and 2) the *bottleneck* objective, which minimizes the maximal distance.

When the metric is the line, then this problem can be viewed as equivalent to the following shoe store problem: in the shoe store problem, the owner has a set of shoes of various sizes and will sell them to to customers who will arrive sequentially, where each customer finishes their transaction before the next customer arrives. If the owner tries to minimize the average difference in shoe size between a shoe and its buyer, then the owner is using the minimum weight objective. If the owner instead tries to minimize the maximal difference, then the owner is using the bottleneck objective.

This problem has been studied most for the general metric, however significant work remains to be done even in the special case of the line metric. Despite the problem having a very 'simple' description, there is still a large gap between the effectiveness of the best known algorithms and the established lowerbound for the effectiveness of any algorithm on this problem. In fact, Koutsoupias and Nanavati in [8] consider the line-metric to be the most interesting, citing reasons such as the offline-version being a trivial problem, the relation to the well-studied cow-path problem, and the applications of this problem in web-markets.

A. Formal Definitions

Formally, for any given instance of the problem I we will denote the set of n servers by $S \subset \mathbb{R}$ and the sequence of requests by $R = (r_1, ..., r_n)$. We will define a completed matching between the set of servers and the requests to be a bijective function $f : R \to S$. Hence for the *bottleneck objective* the goal is to construct a function f such that the *cost* of f defined by $c(f) = \max_{i \in \{1,...,n\}} |f(r_i) - r_i|$ is minimized, whereas for the minimum weight objective the the goal is to construct f such that the cost $c(f) = \sum_{i=1}^{n} |f(r_i) - r_i|$ is minimized.

Competitive analysis is used to determine the effectiveness of algorithms for online problems. Define $\mathcal{A}$ to be the set of all algorithms on for the problem and define $\mathcal{I}$ to be the set of all possible instances. Because the method of analysis for deterministic and randomized algorithms is fundamentally different, we define $\mathcal{A}_\mathcal{D} \subset \mathcal{A}$ to be the set of deterministic algorithms. The *competitive ratio* ρ for a deterministic algorithm $A \in \mathcal{A}_\mathcal{D}$ on an online problem is given by

$$\rho = \max_{I \in \mathcal{I}} \left(\frac{\mathrm{A}[I]}{\textsc{Opt}[I]} \right),$$

where $A[I]$ and $\textsc{Opt}[I]$ denote the cost of the algorithm and the cost of the optimal offline solution respectively. Since we will be looking at the competitive ratio of randomized algorithms, the competitive ratio will be evaluated in expectation, specifically for any randomized algorithm $A \in \mathcal{A}$ the competitive ratio is defined to be

$$\rho = \max_{I \in \mathcal{I}} \left(E \left[\frac{A[I]}{\textsc{Opt}[I]} \right] \right).$$

For notational convenience, when we say that an algorithm is $h(n)$-*competitive*, then we mean that the competitive ratio ρ for that algorithm is no better than $h(n)$ on instances with n servers.

To establish lowerbounds for the competitive ratios of these problems, we use Yao's minimax principle. Yao's principle implies that for any probability distributions p over $\mathcal{A}_\mathcal{D}$ and q over $\mathcal{I}$, where A is some algorithm chosen according to p and I is some instance chosen according to q, then

$$\max_{i \in \mathcal{I}} E\left[A\left[i\right]\right] \geq \min_{a \in \mathcal{A}_\mathcal{D}} E\left[a\left[I\right]\right].$$

Hence by defining c_a^p as the smallest competitive ratio of the algorithm a under the distribution p, Yao's principle implies that

$$\rho(A) \geq \min_{a \in \mathcal{A}_\mathcal{D}} c_a^p,$$

where $\rho(A)$ denotes the competitive ratio of A.

B. Previous Results

The natural greedy algorithm, which sends each incoming request to its closest available server, has been shown to be no better than $(2^n - 1)$-competitive on the line metric for both the bottleneck and minimum weight objectives [6]. For the minimum weight objective, Kalyanasundaram and Pruhs in [6] and Khuller, Mitchell, and Vazirani in [7] gave a $(2n-1)$-competitive deterministic algorithm PERMUTATION. For the general metric, Meyerson, Nanavati, and Poplawski gave a $O(\log^3 n)$-competitive randomized algorithm in [9], which was improved upon by Bansal, Buchbinder, Gupta, and Naor in [2] to give a $O(\log^2 n)$-competitive randomized algorithm, but there still remains a gap to the established lower bound of $\Omega(\log n)$ (from the same paper). In the case where the metric is specifically the line, the Work Function Algorithm was shown to have competitive ratio $O(n)$ in [8], but this upper bound was improved by Antoniadis, Barcelo, Nugent, Pruhs, and Scquizzato who gave a $o(n)$-competitive deterministic algorithm on the line in [1]. The best known randomized algorithm, HARMONIC, was presented by Gupta and Lewi and shown to be $O(\log n)$-competitive in [4]. However, only a lower bound of 9.001 has been established for deterministic algorithms on the line [3] and no lower bound has been established for the randomized algorithms on the line.

The best known deterministic algorithm for the bottleneck objective is also PERMUTATION and has been shown to be $(2n-1)$-competitive in [6]. The highest known lowerbound for the competitive ratio for deterministic algorithms was presented by Idury and Schaffer to be $1.5n$ in [5]. To the best of our knowledge, there is no established lowerbound for the competitive ratio on the bottleneck objective for randomized algorithms.

C. Our Results

We present lowerbounds of the competitive ratios for both the minimum weight and bottleneck objectives. For the minimum weight objective, the gap is improved to a lowerbound of 2 and a previously established upperbound of $\log(n)$ from the algorithm HARMONIC. For the bottleneck objective, the gap in the competitive ratio is improved to a lowerbound of $\frac{n}{2}$ and an already established upperbound of $2n-1$ from the algorithm PERMUTATION. Because this shows that the lowerbound of the competitive ratio of any randomized algorithm on the bottleneck objective is higher than the competitive ratio of the best known algorithm for the minimum weight objective, we have also proven that the bottleneck objective is a *harder* problem for randomized algorithms than the minimum weight objective on the line.

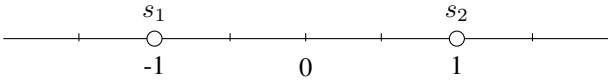
II. MININIMUM WEIGHT OBJECTIVE

A. A Minimum Weight Lowerbound

First, we will present a simple lowerbound of 2 for the competitive ratio of any randomized algorithm on the minimum weight objective.

Theorem 1. The lower bound of any randomized algorithm on the Min-Weight objective is 2.

Proof. To achieve this, we only need 2 servers:



In this instance, the first request will arrive at 0 and the second request will arrive at either -1 or 1 with uniform probability. Hence, the optimal offline assignment will be to match the second request with whichever server it lands on and the first request with the other. This makes the optimal total weight a constant 1. However, any algorithm will have a 50% chance to match the first request with the server at the location of the second request. This means that any algorithm will have an expected cost of $\frac{1}{2}(1) + \frac{1}{2}(3) = 2$. Hence, the competitive ratio of any randomized algorithm must be at least 2.

□

This lowerbound unfortunately does not increase easily. It can be shown that increasing the number of servers while still maintaining the same format of the instance does not increase the lowerbound. See the full version of this work for details.

B. Reducing The Set Of Algorithms For Minimum Weight

One way to reduce the problem of constructing lower bounds is to reduce the possible search pool of algorithms you are evaluating for example. Recall that we construct lower bounds by consider the value of

$$\min_{A \in \mathcal{A}} \left(E\left[\frac{A[I]}{\text{OPT}[I]} \right] \right)$$

for any specific instance I. Hence, if you can prove that

$$\min_{A \in \mathcal{A}} \left(E\left[\frac{A[I]}{\text{OPT}[I]} \right] \right) = \min_{A \in \mathcal{A}_\mathcal{P}} \left(E\left[\frac{A[I]}{\text{OPT}[I]} \right] \right)$$

for some $\mathcal{A}_\mathcal{P} \subset \mathcal{A}$, then you are effectively reducing the type of algorithms which you are minimizing over.

Theorem 2. Given any instance of the problem for the minimum weight objective, there exists an optimal min-weight matching function $f : R \to S$ that satisfies $f(r_j) \in N_f(r_j)$, where $N_f(r_j)$ denotes the neighboring unmatched servers of r_j on the line. Formally, by letting

$$\begin{aligned} S_j &= S \setminus \{f(r_1), ..., f(r_{j-1})\}, \\ L_j &= \max\{s | s \in S_j, s \leq r_j\}, \text{ and} \\ R_j &= \min\{s | s \in S_j, s \geq r_j\}, \end{aligned}$$

then $N_f(r_j) = \{L_j, R_j\}$.

Proof. For the sake of contradiction, we will suppose there does not exist a matching that matches each request r_j to L_j or R_j. Let f denote any optimal matching that satisfies the above property up to some point. That is, $f(r_i) \in N_f(r_i)$ for all i up to some $n \leq k$, but $f(r_j) \notin N_f(r_j)$. We will use the following notation for the rest of the proof:

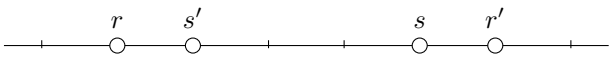
$$r = r_j, \quad s = f(r_j), \quad s' = R_j, \quad r' = f^{-1}(s').$$

We will also let $\tilde{f}$ denote the matching that sends r to s' and r' to s and $\tilde{f}(r_j) = f(r_j)$ for all other requests r_j.

To see there are only 8 cases which need to be considered, note that we must have that $r < s'$, but cannot have that $r < s < s'$, since both s and s' must be available at the time of arrival for the jth request but s' is defined to be the closest right server. Thus, we have 3 possible cases where $r = \min\{r, s, s', r'\}$, 2 cases where $r' = \min\{r, s, s', r'\}$, and 3 cases where $s = \min\{r, s, s', r'\}$ (s' can't ever be the leftmost due to the restriction $r < s'$).

This gives us the following 8 possible cases (note that the figures are not necessarily drawn to scale, they are just to display ordering):

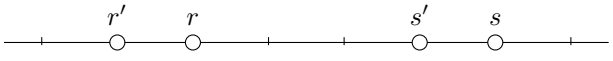
Case 1



Case 2

Case 3

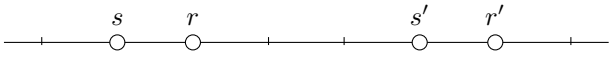
Case 4



Case 5

Case 6

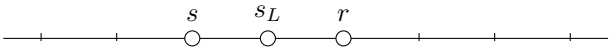
Case 7



Case 8

In case 1, case 2, case 5, and case 6, the cost of $\tilde{f}$ will actually be less than the cost of f, since $|r - f(r)| > |r - \tilde{f}(r)|$ and $|r' - f(r')| > |r' - \tilde{f}(r')|$. In cases 3 and 4, the cost of $\tilde{f}$ will be equal to the cost of f. Finally, in cases 7 and 8, we see that the cost of $\tilde{f}$ will actually be bigger than the cost of f. But this is not a problem - first, if $s = L_j$ (if s is r's closest left available server), then we have a contradiction because we originally supposed that r was not matched with either of its left or right neighboring servers. Thus, letting s_L denote L_j, we must have the following order:

Case 8



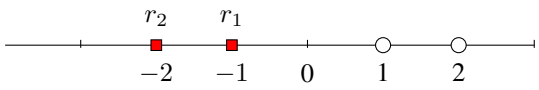
But now, we can see that the mirror of either case 1, 2, 3, or 4 must apply, since these are the cases where $r < s' < s$, and thus we know that constructing a new matching $\tilde{f}'$ such that $\tilde{f}'(r) = s_L$ and $\tilde{f}'^{-1}(s) = \tilde{f}'^{-1}(s_L)$ (where all other requests correspond to f) will satisfy that the total cost of $\tilde{f}'$ will be less than or equal to the total cost of f.

In all of these cases, we have constructed a new matching whose total cost is less than or equal to f where r is matched to one of its closest available neighbors. This is a contradiction with respect to the construction of f and completes the proof. □

Hence, when analyzing the lowerbound of the competitive ratio on a specific instance, we can restrict our search space to algorithms which always matching to either the left or right closest servers.

III. Bottleneck Objective

Unfortunately, Theorem 2 doesn't hold for the bottleneck objective. To see this, consider an instance of just two servers:

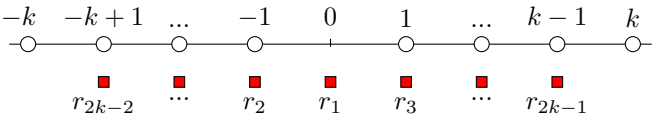


Now, if Theorem 2 were also true for the bottleneck objective, then we would obtain an optimal matching by assigning r_1 to 1 and r_2 to 2. However, this will result in a maximal distance of $2 - r_2 = 4$. But if instead we match r_1 to 2 and r_2 to 1, then the maximal distance is instead 3. This is a contradiction to the optimality of the first matching, confirming that Theorem 2 does not hold for the bottleneck objective.

However, without reducing the search pool of algorithms we are still able to prove a linear lowerbound for the bottleneck objective.

Theorem 3. The competitive ratio of any randomized algorithm for the bottleneck objective with n servers is no better than $\frac{n}{2} = \Omega(n)$.

Proof. We will construct an instance with $n = 2k$ servers, where the servers are located at $\{-k, -k+1, ..., -1, 1, ..., k-1, k\}$ and the requests will arrive at $r_1 = 0, r_2 = -1, r_3 = 1, r_4 = -2, r_5 = 2, ..., \ r_{2k-2} = -k+1$, and $r_{2k-1} = k-1$ with the final request r_{2k} landing either on $-k$ or k with uniform probability.



Note that after r_{2k-1} has been matched by any algorithm, there will always be exactly one server remaining, which we will denote by $s \in \mathbb{Z}$. The expected cost of the bottleneck edge in this algorithm will be at least the expected cost of the last edge created with s, since the maximal distance pairing will necessarily be bigger than or equal to $d(r_{2k}, s) = |r_{2k} - s|$. Since $s \geq -k$, the edge cost if $r_{2k} = -k$ will be $k + s$. If $r_{2k} = k$, then the edge cost will be $k - s$, since $s \leq k$. Hence, the expected cost of any algorithm will be at least $\frac{1}{2}((k+s) + (k-s)) = k$.

The offline optimal matching, however, would have a maximal pairing cost of 1. To achieve this, simply match the left most available server with the left most unassigned request and repeat. If $r_{2k} = k$, then all the nonpositive requests will have a distance of 1 to their respective server while all the positive requests will have a cost of 0. Thus the competitive ratio of any algorithm will be no less than $\frac{k}{1} = k$, where k is half the number of servers used. □

Thus for the bottleneck objective on n servers, the gap for randomized algorithms is between $\frac{n}{2}$ and $2n - 1$.

IV. Concluding Remarks

Despite having shown that the lowerbounds of the competitive ratios for the minimum weight objective is 2 and the bottleneck objective is $\frac{n}{2}$ for a set of n servers, there is still work to be done to close the gap to the upperbounds of both problems. The algorithm HARMONIC is still the best known algorithm for the minimum weight objective while still being $\log(n)$ competitive, while it still hasn't even been proven that there is no c-competitive algorithm for some constant c. For the bottleneck objective, the gap is now reduced to a lowerbound of $\frac{n}{2}$ to the upperbound of $2n - 1$ from the algorithm PERMUTATION. While the gap is still open, it is now clear that the bottleneck objective is a "harder" problem than the minimum weight objective on the line, since the lowerbound for the competitive ratio of any algorithm on the bottleneck objective is greater than the competitive ratio of HARMONIC for the minimum weight objective.

References

[1] A. Antoniadis, N. Barcelo, M. Nugent, K. Pruhs, and M. Scquizzato. A o(n)-competitive determinstic algorithm for online matching on a line. In *WAOA14*, page to appear, 2014.

[2] N. Bansal, N. Buchbinder, A. Gupta, and J. Naor. A randomized $o(\log^2 k)$-competitive algorithm for metric bipartite matching. *Algorithmica*, 68(2):390–403, 2014.

[3] B. Fuchs, W. Hochstättler, and W. Kern. Online matching on a line. In *Electronic Notes In Discrete Mathematics*. Elsevier, 2003.

[4] Anupam Gupta and Kevin Lewi. The online metric matching problem for doubling metrics. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, volume 7391 of *Lecture Notes in Computer Science*, pages 424–435. Springer Berlin Heidelberg, 2012.

[5] R. Idury and A. A. Schäffer. A better lower bound for on-line bottleneck matching. http://www.ncbi.nlm.nih.gov/core/assets/cbb/files/Firehouse.pdf. Accessed: 2015-09-7.

[6] B. Kalyanasundaram and K. Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993.

[7] S. Khuller, S. G. Mitchell, and V. V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127:251–264, 1994.

[8] E. Koutsoupias and A. Nanavati. The online matching problem on a line. In *WAOA03*, pages 179–191, 2003.

[9] A. Meyerson, A. Nanavati, and L. J. Poplawski. Randomizedi online algorithms for minimum metric bipartite matching. In *SODA 06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 954–959, 2006.