Minimum Cost Perfect Matching with Delays for Two Sources

Yuval Emek[1], Yaacov Shapiro[2], and Yuyi Wang[3]

[1] Technion, Israel. `yemek@technion.ac.il`
[2] Technion, Israel. `shapiro.yaacov@gmail.com`
[3] ETH Zürich, Switzerland. `yuwang@ethz.ch`

Abstract. We study a version of the online *min-cost perfect matching with delays (MPMD)* problem recently introduced by Emek et al. (STOC 2016). In this problem, requests arrive in a continuous time online fashion and should be matched to each other. Each request emerges from one out of n *sources*, with metric inter-source distances. The algorithm is allowed to delay the matching of requests, but with a cost: when matching two requests, it pays the distance between their respective sources and the time each request has waited from its arrival until it was matched. In this paper, we consider the special case of $n = 2$ sources that captures the essence of the match-or-wait challenge (cf. rent-or-buy). It turns out that even for this degenerate metric space, the problem is far from trivial. Our results include a deterministic 3-competitive online algorithm for this problem, a proof that no deterministic online algorithm can have competitive ratio smaller than 3, and a proof that the same lower bound applies also for the restricted family of memoryless randomized algorithms.

1 Introduction

The abundance of hand held devices and the ease of application development is fueling the increasing popularity of online games. With that, the demand for head-to-head competition or players teaming up to complete a mission is growing fast. Ranging from classic games like Chess to car racing games like Asphalt 8, users wish to be matched with suitable opponents. The online gaming platform tries to find a match for each player while conflicted between two desired criteria: find a worthy opponent and find it fast. It is not hard to imagine a situation where the only available opponents are a poor match either due to a large difference between the players' skills and experience or due to the network distance between the players which may lead to significant communication delays. Should the platform wait, risking bored (and thus, dissatisfied) users? For how long?

Emek et al. [8] recently formalized this challenge in terms of the *min-cost perfect matching with delays (MPMD)* problem. In this problem requests arrive in an online fashion at the points of a finite metric space (known in advance). The online algorithm serves the requests by matching them to each other (i.e., partitioning the request set into pairs), where each match incurs a space cost equal to the metric distance between the locations of the matched requests. The crux of this problem is that it is not mandatory to serve the requests immediately; rather, the online algorithm is allowed to delay its matching commitments, but this incurs an additional time cost.

Some online gaming platforms will match a pending human player with a virtual (computer) opponent if a suitable human opponent cannot be found within a reasonable time frame. On the positive side, this allows the platform to shorten the waiting times of its users, but on the negative side, a player matched with a virtual opponent may be slightly disappointed: after all, it is more enjoyable to compete with your peers. This imposes an additional algorithmic challenge on behalf of the gaming platform: For how long should the platform wait before it matches a pending user to a virtual opponent? The challenge faced by a gaming platform that is allowed to match a human player to a virtual opponent is captured by a variant of the MPMD problem, referred to in [8] as *MPMDfp*, where the algorithm can serve a pending request without matching it to another request, paying a fixed penalty (hence the 'fp' abbreviation). Among other results, Emek et al. showed that the MPMDfp problem on an n-point metric space can be reduced to the MPMD problem on a $(2n)$-point metric space.

In this paper, we focus on a special case of the MPMD problem, referred to as *2-MPMD*, where the metric space consists of only two points with a unit distance between them, that is, the requests emerge from one of two possible *sources*. It turns out that even for this degenerate metric space, the problem is far from trivial. Moreover, the 2-MPMD problem generalizes the MPMDfp problem on a single point that corresponds to an online game in which all (user-to-user) matches are considered equally good and it is only the waiting times and penalties paid for matching users with a virtual opponent that affect the platform's total cost. This problem is interesting by its own right as it captures the essence of the wait-or-match question (cf. rent-or-buy) while abstracting away the space cost component arising from the distances in the metric space.

1.1 Model

An instance of the *2-source minimum cost perfect matching with delays (2-MPMD)* problem consists of two *sources* (denoted a and b) and a set R of *requests*. Each request $r \in R$ is characterized by its source $x(r) \in \{a, b\}$ and its *arrival time* $t(r) \in \mathbb{R}_{\geq 0}$. The request set R is provided to the algorithm in a continuous time online fashion so that request $r \in R$ is reported at time $t(r)$.

Assume throughout that $|R|$ is even. The output of the algorithm is a perfect matching of R, namely, a partition of R into unordered request pairs. Though this is computed online, the algorithm is allowed (and often required) to delay the matching of any request in R. This delay comes with a cost: the time between the arrival of a request and it being matched is added to the cost incurred by the algorithm.

Specifically, given two requests $r_1, r_2 \in R$, we denote a *match* operation $m(r_1, r_2, t)$ as the assignment of r_1 and r_2 into an unordered pair at time $t \geq max\{t(r_1), t(r_2)\}$. Match m incurs two types of costs on each request: *space cost* and *time cost*. If r_1 and r_2 have different sources (i.e., $x(r_1) \neq x(r_2)$), referred to hereafter as *matching across*, then the space cost for each of the requests is $1/2$, otherwise it is 0 (this choice of space cost convention reflects the implicit assumption that there is a unit distance between the two sources). The time costs for requests r_1 and r_2 are $t - t(r_1)$ and $t - t(r_2)$, respectively. The total cost of match operation m is denoted $\text{cost}(m)$ and is defined to be the sum of the space and time costs of the two involved requests. For algorithm ALG, we denote the set of unordered pairs that have been produced by it as M_{ALG} and the total cost incurred by the algorithm, denoted by $\text{cost}_{ALG}(R)$, is defined to be $\text{cost}_{ALG}(R) = \sum_{m \in M_{ALG}} \text{cost}(m)$.

When request r arrives, it is initially referred to as *open*; once the algorithm has matched it with another request, it becomes *matched*. Notice that this definition is only relevant in the context of a certain algorithm since two different algorithms may have matched the requests differently or at different times.

Our goal is to minimize the total cost of the match operations that our algorithm performs. Adhering to the common practice in the theory of online algorithms [4], the quality of the algorithmic solutions is measured in terms of their *competitive ratio*: Online algorithm ALG is said to be α-*competitive* if there exists a universal constant β such that $\text{cost}_{ALG}(R) \leq \alpha \cdot \text{cost}_{OPT}(R) + \beta$ for every (even size) request sequence R, where OPT is an optimal offline algorithm (the cost is taken in expectation if ALG is randomized). We notice that ALG has no a priori knowledge of R.

1.2 Related Work

Matching is a classic problem in graph theory and combinatorial optimization since the seminal work of Edmonds [7,6]. The matching problem has been studied in the context of online computation as well, starting with the classic paper of Karp, Vazirani, and Vazirani [11] that ignited the interest in online matching and attracted a lot of attention to the different versions of this problem [10,12,14,15,3,9,1,5,13,16,17]. In these online versions, it is usually assumed that the requests belong to one side of a bipartite graph whose other side is given in advance.

Emek et al. [8] recently introduced the MPMD problem which differs from the previously studied online matching versions in that the underlying graph (or metric space) is known in advance and the algorithmic challenge stems from the unknown locations and arrival times of the requests (whose number is unbounded). They present a randomized algorithm with competitive ratio $O(\log^2 n + \log \Delta)$, where n is the number of points in the metric space and Δ is the aspect ratio. Wang and Wattenhofer [18] show that the algorithm of [8] can be modified to treat the bipartite version of the MPMD problem (left outside the scope of the present paper), obtaining the same competitiveness. Azar et al. [2] devise another online MPMD algorithm with an improved logarithmic competitive ratio. They also prove that no (randomized) online MPMD algorithm can have competitive ratio better than $\Omega(\sqrt{\log n})$ in the all-pairs version and $\Omega(\log^{1/3} n)$ in the bipartite version.

1.3 Our Contribution

The general case of the online MPMD problem received a lot of attention in the last year, narrowing the asymptotic gap between the upper and lower bounds on the competitiveness of this problem [8,18,2]. The algorithms presented in [8] and [2] clearly imply a constant upper bound on the competitiveness of the 2-MPMD problem, but the analyses in these papers are not necessarily tailored to optimize this constant: the upper bound guaranteed by the analysis in [2] is 5; the upper bound guaranteed by the analysis in [8] is even larger, however, examining [8]'s randomized online algorithm (its 2-MPMD restriction) more carefully reveals that its competitive ratio is at most 3. Is this optimal? Can one ensure the same upper bound with a deterministic online algorithm?

In this paper, we answer these two questions on the affirmative. First, in Section 2, we introduce a deterministic variant of the online algorithm of [8] (restricted to the special case of the 2-MPMD problem) and establish an upper bound of 3 on its competitive ratio. Then, in Section 3, we prove that any deterministic online 2-MPMD algorithm must have a competitive ratio of at least 3. Finally, in Section 4, we investigate the competitiveness of *memoryless* online 2-MPMD algorithms — a family of randomized algorithms that include the algorithm of [8] (refer to Section 4 for an exact definition) — proving that it is at least 3 as well. While the upper bound of Section 2 clearly holds for the MPMDfp problem on a single point (recall that this is a special case of 2-MPMD), it is interesting to point out that the lower bounds of Sections 3 and 4 also hold for that problem.

2 An Online 2-MPMD Algorithm

In this section we present a deterministic online 2-MPMD algorithm (Section 2.1), referred to as *delayed matching on 2 sources (DM2)*, and prove that its competitive ratio is 3 (Section 2.2). A matching lower bound will be established in Section 3.

2.1 Algorithm DM2

In this section we present our online algorithm $DM2$. While $DM2$ is designed for a continuous time environment, it is more easily understood when described as if it was operating in a discrete time environment, taking discrete time steps. The time difference dt between two consecutive steps is taken to be infinitesimally small so that we can assume without loss of generality that every request arrives in a separate time step.

The algorithm holds a counter T initialized to 0. For a given time step t and a request r_1 arriving during that time step, if there exists another open request r_2 in the same source, then $DM2$ matches the two requests immediately. If during this time step, there are two open requests in different sources (arriving in this time step or in previous ones), then $DM2$ will increase T by dt. When T reaches τ, $DM2$ matches across and resets T back to 0. Refer to Algorithm 1 for a pseudocode.

Algorithm 1 Algorithm $DM2$ at time step t.

```
1: if there exist two open requests $r_1 \neq r_2$ with $x(r_1) = x(r_2)$ then
2:     $match(r_1, r_2)$
3: else if there exist two open requests $r_1 \neq r_2$ with $x(r_1) \neq x(r_2)$ then
4:     $T \leftarrow T + dt$
5:     if $T = \tau$ then
6:         $match(r_1, r_2)$
7:         $T \leftarrow 0$
8:     end if
9: end if
```

2.2 Analysis

Our goal in this section is to analyze the competitiveness of the online algorithm presented in Section 2.1, establishing the following theorem.

Theorem 1. *Algorithm $DM2$ is 3-competitive.*

We say that an (online or offline) algorithm A is *smart* if it satisfies the following property: If request r arrives at a source where there already exists an open request $r' \neq r$, i.e., $x(r') = x(r)$ and $t(r') < t(r)$, then A matches r and r' immediately, that is, at time $t(r)$. Notice that any smart algorithm will never have more than two open requests for a positive duration of time. Algorithm $DM2$ is clearly smart by definition.

Lemma 1. *There exists an online method that transforms any algorithm A into a smart algorithm $\tilde{A}$ without increasing the total cost incurred by the algorithm.*

Proof. Let R be the request sequence and consider the first occurrence of two open requests r_1, r_2 with $x(r_1) = x(r_2)$ and $t(r_1) < t(r_2)$ that algorithm A does not match immediately (i.e., at time $t(r_2)$). We construct an algorithm A' that behaves exactly like A up to time $t(r_2)$ and matches r_1 with r_2 at time $t(r_2)$ and show that the cost incurred by A' is not greater than that of A. This argument can then be repeated to turn A into the desired smart algorithm $\tilde{A}$.

Algorithm A' will match r_1 with r_2 at time $t(r_2)$ and continue as follows. If A matches r_1 with r_2 at a later time $t' > t(r_2)$, then all other matching operations of A' are identical to those of A. In this case, $\text{cost}_{A'}(R)$ is clearly smaller than $\text{cost}_A(R)$. The more interesting case is when A matches r_1 to $r'_1 \neq r_2$ at time t_1 and r_2 to $r'_2 \neq r_1$ at time t_2; in this case, A' will match r'_1 with r'_2 at time $max\{t_1, t_2\}$. Again, all other matching operations of A' are identical to those of A.

It remains to prove that $\text{cost}_{A'}(R) \leq \text{cost}_A(R)$ also in this case. To that end, notice that

$$\begin{aligned}
\text{cost}_A(R) - \text{cost}_{A'}(R) &\geq (t_1 - t(r_1) + t_1 - t(r'_1) + t_2 - t(r_2) + t_2 - t(r'_2)) \\
&\quad - (t(r_2) - t(r_1) + 2\max\{t_1, t_2\} - t(r'_1) - t(r'_2)) \\
&= 2\,(t_1 + t_2 - t(r_2) - \max\{t_1, t_2\})\ .
\end{aligned}$$

The last expression is non-negative since we know that $t_1, t_2 \geq t(r_2)$.

We subsequently assume that OPT is smart. The cost incurred by a smart algorithm A (online or offline) is comprised of three *cost components*:
(C1) the space cost incurred by A for matching across;
(C2) the time cost incurred by A while there exists a single open request; and
(C3) the time cost incurred by A while there exist two open requests (one at each source).

Observation 1 *The parity of the number of open requests is the same for $DM2$ and OPT at any time t.*

From Observation 1 we conclude that cost component (C2) for $DM2$ is identical to that of OPT. We therefore ignore it in the subsequent analysis (this can only hurt the upper bound we obtain on the competitive ratio of $DM2$).

Our analysis relies on partitioning the time axis into *phases*. Each phase starts when the previous phase ends (the first phase starts at time 0) and ends when $DM2$ performs a match across (OPT might have open requests at this stage). Note also that a match across can only occur (Line 10) when T has increased from 0 to τ since the last time a match across occurred (i.e., in this phase only).

Let $M_A(P)$ denote the set of request pairs matched by algorithm A during phase P so the cost that algorithm A pays for P is $\text{cost}_A(P) = \sum_{m \in M_A(P)} \text{cost}_A(m)$. Since we ignore cost component (C2), the cost for $DM2$ of any phase P is always $cost_{DM2}(P) = 1 + 2\tau$.

Phase P is said to be *clean* if when it starts, OPT does not have any open requests. Otherwise, P is said to be *dirty*. Phase P is said to be *even numbered* if OPT performs an even number of matches across during the time interval P. Otherwise, P is said to be *odd numbered*.

Observation 2 *When a dirty phase starts, OPT has exactly two open requests.*

Proof. From Observation 1 we know OPT has an even number of open requests; it can not be larger than 2 because OPT is smart and it can not be 0 because the phase is dirty.

Observation 3 *In every phase the number of requests that appear in each source is odd.*

Proof. $DM2$ starts and ends each phase with no open requests, and since it also matches across exactly once during the phase, the number of requests appearing during the phase in each source must be odd.

Observation 4 follows from Observation 3 and from the fact that $DM2$ matches across exactly once (and in particular an odd number of times) in each phase.

Observation 4 *Given two consecutive phases P_1 and P_2, if both are clean or both are dirty, then P_1 must be an odd numbered phase. If one of them is dirty and the other is clean, then P_1 must be even numbered.*

We now turn to define the notion of a *super phase* which consists of a clean phase followed by a maximal (possibly empty) contiguous sequence of dirty phases. This notion uniquely induces a partition of the phase sequence into super phases. We denote the cost that algorithm A pays for super phase S as $\text{cost}_A(S) = \sum_{P \in S} \text{cost}_A(P)$. By definition, when a super phase starts, both $DM2$ and OPT have no open requests. This means that we can establish Theorem 1 by proving the following lemma.

Lemma 2. *There exists a choice of the parameter $\tau > 0$ that guarantees $\text{cost}_{DM2}(S) \leq 3 \cdot \text{cost}_{OPT}(S)$ for every super phase S.*

Proof. Consider some super phase S comprised of a single clean phase followed by $n \geq 0$ dirty phases. We know that $DM2$ pays $1 + 2\tau$ for each phase, so $\text{cost}_{DM2}(S) = (n+1)(1+2\tau)$.

If S consists of exactly one phase P (i.e., $n = 0$), then by Observation 4, P is odd numbered. Therefore, OPT matched across at least once during super phase S, hence $\text{cost}_{OPT}(S) \geq 1$. This proves the assertion under the requirement that $\tau \leq 1$.

Assume hereafter that S contains $n \geq 1$ dirty phases. Refer to the first (clean) phase of S as P_0 and to the subsequent n dirty phases as $P_1, .., P_n$ in order of appearance. By Observation 4, phase P_0 is even numbered, therefore, during this phase, OPT either matched across at least twice or did not match across at all. The former case implies that $\text{cost}_{OPT}(P_0) \geq 2$; in the latter case, we know by the design of $DM2$ that OPT paid at least 2τ in time cost. This means that $\text{cost}_{OPT}(P_0) \geq min\{2, 2\tau\}$.

Since phases $P_1, .., P_n$ are all dirty, Observation 4 ensures that phases $P_1, .., P_{n-1}$ are all odd numbered, hence OPT must have matched across at least once during each one of them. It follows that $\text{cost}_{OPT}(P_i) \geq 1$ for every $1 \leq i \leq n-1$. Therefore, the total cost of OPT for super phase S is $\text{cost}_{OPT}(S) \geq min\{2, 2\tau\} + (n-1)$.

To establish the assertion, we require that $(n+1)(1+2\tau) \leq 3(min\{2, 2\tau\} + (n-1))$. To that end, we let

$$f(\tau) = (n+1)(1+2\tau) - 3(min\{2, 2\tau\} + (n-1)) = 2n\tau + 2\tau - 6min\{1, \tau\} - 2n + 4$$

and require that $f(\tau) \leq 0$. Observing that setting $\tau < 1$ implies

$$f(\tau) = 2n\tau + 2\tau - 6\tau - 2n + 4 = (1-\tau)(4-2n)$$

which means that $f(\tau) > 0$ for $n = 1$, forces τ to be at least 1. Recalling the prior constraint of $\tau \leq 1$, we conclude that τ must be 1. Indeed, by setting $\tau = 1$, we obtain $f(\tau) = 0$ as required.

3 A Lower Bound for Deterministic Algorithms

We now turn to show that the algorithm presented in Section 2.1 is optimal by establishing a matching lower bound.

Theorem 2. *For any $\delta > 0$, no deterministic online 2-MPMD algorithm can have a competitive ratio of $3 - \delta$.*

Theorem 2 is established by proving that for every deterministic online 2-MPMD algorithm A, there exists a request sequence R with arbitrarily large $\text{cost}_{OPT}(R)$ such that $\text{cost}_A(R) \geq (3-\delta)\text{cost}_{OPT}(R)$. The key ingredient in the construction of R is a *gadget* G satisfying $\text{cost}_A(G) \geq (3-\delta)\text{cost}_{OPT}(G)$; the request sequence R is then constructed by repeatedly introducing instances of this gadget with sufficiently large time gaps between consecutive copies.

Gadget G is comprised of $2n$ requests, denoted by $r_1, \ldots r_n$ and $r'_1, \ldots, r'_n$, with $x(r_i) = a$, $x(r'_i) = b$, and $t(r_i) = t(r'_i)$ for every $1 \leq i \leq n$. Requests r_1 and r'_1 are the only requests certain to appear, while the appearance of the rest of the requests r_i, r'_i depends on the behavior of A. Given that G includes requests r_i and r'_i (i.e., $n \geq i$), let t_i be the difference (in absolute value) between time $t(r_i)$ and the time when A performed $m(r_i, r'_i)$ (the time difference t_i is well defined since A must eventually match $m(r_i, r'_i)$ as otherwise its competitive ratio is unbounded). The appearance of requests r_{i+1} and r'_{i+1} then abides the following rule: if $t_i < 1$, then requests r_{i+1} and r'_{i+1} are introduced at time

$$t(r_{i+1}) = t(r'_{i+1}) = t(r_i) + t_i + \epsilon$$

for a sufficiently small $\epsilon > 0$; otherwise, the gadget ends (i.e., $n = i$). This holds until the first odd $i \geq 3$ that fulfills $\frac{i-2}{i} \geq 1 - \delta$ after which there will appear no more requests (i.e., $n = i$).

Lemma 3. *For every $\delta > 0$, the construction of G ensures that $\text{cost}_A(G) \geq (3-\delta)\text{cost}_{OPT}(G)$.*

Proof. If $n = 1$, then $\text{cost}_A(G) = 1 + 2t_1$ whereas $\text{cost}_{OPT}(G) = 1$, thus $\frac{\text{cost}_A(G)}{\text{cost}_{OPT}(G)} = \frac{1+2t_1}{1}$ which is at least 3 since $t_1 \geq 1$. If $n = 2$, then $\text{cost}_A(G) = 2 + 2t_1 + 2t_2$ whereas $\text{cost}_{OPT}(G) = 2t_1$ since OPT performs $\{m(r_1, r_2), m(r'_1, r'_2)\}$. Therefore, $\frac{\text{cost}_A(G)}{\text{cost}_{OPT}(G)} = \frac{2+2t_1+2t_2}{2t_1} = \frac{2}{2t_1} + \frac{2t_1}{2t_1} + \frac{2t_2}{2t_1}$ which is at least 3 since $t_1 < 1 \leq t_2$.

Assume hereafter that $n \geq 3$ is odd (the case of even $n \geq 3$ is similar and deferred to the full version). Notice that $\text{cost}_A(G)$ is always $n+2t_1+2t_2+\cdots+2t_n$ whereas OPT can choose between the following options (among others).

1. Perform a match across $m(r_1, r'_1, t(r_1))$ for the first two requests; match all other requests by performing $m(r_{2j}, r_{2j+1}, t(r_{2j+1}))$ and $m(r'_{2j}, r'_{2j+1}, t(r'_{2j+1}))$ for every $1 < j \leq (n-1)/2$.
   Ignoring ϵ-terms that can be made arbitrarily small, this results in $\text{cost}_{OPT}(G) = 1 + 2t_2 + 2t_4 + \cdots + 2t_{n-1}$.
2. Perform a match across $m(r_n, r'_n, t(r_n))$ for the last two requests; match all other requests by performing $m(r_{2j-1}, r_{2j}, t(r_{2j}))$ and $m(r'_{2j-1}, r'_{2j}, t(r'_{2j}))$ for every $1 \leq j \leq (n-1)/2$.
   Ignoring ϵ-terms that can be made arbitrarily small, this results in $\text{cost}_{OPT}(G) = 1 + 2t_1 + 2t_3 + \cdots + 2t_{n-2}$.

Denoting $T_{odd} = \sum_{j=1}^{\lfloor n/2 \rfloor} t_{2j-1}$ and $T_{even} = \sum_{j=1}^{\lceil n/2 \rceil -1} t_{2j}$ implies that $\text{cost}_{OPT}(G)$ can never exceed $\min\{1 + 2T_{even}, 1 + 2T_{odd}\}$. Since $t_i < 1$ for every $1 \leq i \leq n-1$, it follows that $\text{cost}_{OPT}(G) < n$.

Consider the case where $t_i < 1$ for every $1 \leq i \leq n-1$ and $t_n \geq 1$. We examine the ratio of $\text{cost}_A(G)$ and $\text{cost}_{OPT}(G)$ to conclude that

$$
\begin{aligned}
&\frac{\text{cost}_A(G)}{\text{cost}_{OPT}(G)} \\
&= \frac{n + 2T_{odd} + 2T_{even} + 2t_n}{1 + 2\min\{T_{even}, T_{odd}\}} \\
&= \frac{1 + 2T_{odd}}{1 + 2\min\{T_{even}, T_{odd}\}} + \frac{1 + 2T_{even}}{1 + 2\min\{T_{even}, T_{odd}\}} + \frac{n - 2 + 2t_n}{1 + 2\min\{T_{even}, T_{odd}\}} \\
&> \frac{1 + 2T_{odd}}{1 + 2T_{odd}} + \frac{1 + 2T_{even}}{1 + 2T_{even}} + \frac{n}{n} = 3\,.
\end{aligned}
$$

It remains to consider the case where $t_i < 1$ for every $1 \leq i \leq n$ which means that n is odd and that $\frac{n-2}{n} \geq 1 - \delta$. Again, we examine the ratio of $\text{cost}_A(G)$ and $\text{cost}_{OPT}(G)$ to conclude that

$$
\begin{aligned}
&\frac{\text{cost}_A(G)}{\text{cost}_{OPT}(G)} \\
&= \frac{n + 2T_{odd} + 2T_{even} + 2t_n}{1 + 2\min\{T_{even}, T_{odd}\}} \\
&= \frac{1 + 2T_{odd}}{1 + 2\min\{T_{even}, T_{odd}\}} + \frac{1 + 2T_{even}}{1 + 2\min\{T_{even}, T_{odd}\}} + \frac{n - 2 + 2t_n}{1 + 2\min\{T_{even}, T_{odd}\}} \\
&\geq \frac{1 + 2T_{odd}}{1 + 2T_{odd}} + \frac{1 + 2T_{even}}{1 + 2T_{even}} + \frac{n-2}{n} \geq 3 - \delta\,.
\end{aligned}
$$

The assertion follows.

4 Memoryless Online Algorithms

We now turn our attention to randomized algorithms. As the deterministic algorithm presented in Section 2 is 3-competitive and the lower bound established in Section 3 states that this cannot be improved, it is natural to ask whether a randomized 2-MPMD algorithm can have competitive ratio smaller than 3. While we dot know the answer to this question in the general case yet, the current section resolves it on the negative for a restricted family of randomized 2-MPMD algorithms.

Recall the notion of smart algorithms presented in Section 2.2. Lemma 1 guarantees that for the sake of establishing negative results, it suffices to consider the class of smart online algorithms as any algorithm can be transformed into a smart algorithm without increasing the cost (since the transformation works in an online fashion, the lemma applies to randomized algorithms as well).

Consider some randomized smart 2-MPMD algorithm ALG. By the definition of smart algorithms, ALG is fully characterized by the parameter $\lambda(t) \in \mathbb{R}_{\geq 0}$, $t \geq 0$, defined so that if there is an open request in each source throughout the infinitesimally small time interval $[t - dt, t)$, then ALG matches across at time t with probability $\lambda(t)dt$. (Strictly speaking, $\lambda(t)$ can also take the special value $1/dt$ in which case $\lambda(t)dt = 1$; in particular, the algorithm is deterministic if $\lambda(t)$ is either 0 or $1/dt$ for every $t \geq 0$.) We say that algorithm ALG is *memoryless* if there exists some $\lambda > 0$ such that $\lambda(t) = \lambda$ for every $t \geq 0$, namely, the probability that ALG matches across at time t depends only on the infinitesimally small time interval $[t - dt, t)$ and is independent of the rest of the history.

Interestingly, the restriction of the randomized online MPMD algorithm of [8] to metric spaces with 2 sources is memoryless with parameter $\lambda = 1$. Although the authors of [8] did not attempt to optimize the (constant) competitive ratio of their algorithm for that special case, a careful examination of the arguments used in the analysis of this algorithm reveals that its competitive ratio is 3. In this section, we show that this cannot be improved.

Theorem 3. *If ALG is a (randomized) memoryless 2-MPMD algorithm with parameter $\lambda \neq 1$, then its competitive ratio is greater than* 3.

Proof. Consider first the case where $\lambda < 1$. Let R be the request sequence consisting of $2n$ requests $r_1, \ldots, r_n$ and $r'_1, \ldots, r'_n$ for some arbitrarily large n so that
(1) $x(r_i) = a$ and $x(r'_i) = b$ for every $1 \leq i \leq n$;
(2) $t_1 = t(r_1) = t(r'_1) = 0$; and
(3) $t_{i+1} = t(r_{i+1}) = t(r'_{i+1}) = t_i + z$ for some sufficiently large real z for every $1 \leq i \leq n - 1$.
Given that $z > 1$, OPT will perform $m(r_i, r'_i, t_i)$ for every $1 \leq i \leq n$ for a total cost of $\text{cost}_{OPT}(R) = n$.

Assuming that all previous requests are already matched by the time r_i and r'_i arrive, ALG will perform either (i) $m(r_i, r'_i, t_i + Y_i)$ for some $0 \leq Y_i < z$; or (ii) $m(r_i, r_{i+1}, t_{i+1})$ and $m(r'_i, r'_{i+1}, t_{i+1})$, recalling that $t_{i+1} = t_i + z$. By the definition of a memoryless algorithm, the probability that (ii) occurs is

$$\mathbb{P}(\text{Exp}(\lambda) > z) = e^{-\lambda z},$$

where $\text{Exp}(\lambda)$ is an exponential random variable with rate parameter λ. Moreover, Y_i is a random variable that behaves like $\text{Exp}(\lambda)$, truncated at z, namely, $Y_i \sim \min\{\text{Exp}(\lambda), z\}$. Standard calculation (see, e.g., the proof of Lemma 4.2 in [8]) then yields that

$$\mathbb{E}(Y_i) = \frac{1}{\lambda}(1 - e^{-\lambda z}).$$

It follows that

$$\lim_{z \to \infty} \mathbb{E}(\text{cost}_{ALG}(R)) = n(1 + 2/\lambda) > 3n,$$

where the last transition follows from the assumption that $\lambda < 1$.

Suppose that $\lambda > 1$ and let R be the request sequence consisting of $4n$ requests $r_1, \dots, r_{2n}$ and $r'_1, \dots, r'_{2n}$ for some arbitrarily large n so that
(1) $x(r_i) = a$ and $x(r'_i) = b$ for every $1 \le i \le 2n$;
(2) $t_1 = t(r_1) = t(r'_1) = 0$;
(3) $t_{2i} = t(r_{2i}) = t(r'_{2i}) = t_{2i-1} + \epsilon$ for some sufficiently small real $\epsilon > 0$ for every $1 \le i \le n$; and (4) $t_{2i+1} = t(r_{2i+1}) = t(r'_{2i+1}) = t_{2i} + z$ for some sufficiently large real z for every $1 \le i \le n-1$. Given that $\epsilon < 1$, OPT will perform $m(r_{2i-1}, r_{2i}, t_{2i})$ and $m(r'_{2i-1}, r'_{2i}, t_{2i})$ for every $1 \le i \le n$ for a total cost of $\text{cost}_{OPT}(R) = 2\epsilon n$.

Assuming that all previous requests are already matched by the time r_{2i-1} and r'_{2i-1} arrive, ALG will perform either (i) $m(r_{2i-1}, r'_{2i-1}, t_{2i-1} + y)$ for some $0 \le y < \epsilon$; or (ii) $m(r_{2i-1}, r_{2i}, t_{2i})$ and $m(r'_{2i-1}, r'_{2i}, t_{2i})$, recalling that $t_{2i} = t_{2i-1} + \epsilon$. Taking p to be the probability that (i) occurs, we conclude by the definition of a memoryless algorithm that $p = \mathbb{P}(\text{Exp}(\lambda) < \epsilon) = 1 - e^{-\lambda\epsilon}$, hence

$$\lambda\epsilon(1 - \lambda\epsilon) < \lambda\epsilon - (\lambda\epsilon)^2/2 < p < \lambda\epsilon$$

by standard approximations of the exponential function.

Condition for the time being on the event that (i) occurs and notice that ALG will now perform either (iii) $m(r_{2i}, r'_{2i}, t_{2i} + Y_i)$ for some $0 \le Y_i < z$; or (iv) $m(r_{2i}, r_{2i+1}, t_{2i+1})$ and $m(r'_{2i}, r'_{2i+1}, t_{2i+1})$, recalling that $t_{2i+1} = t_{2i} + z$. As before, the probability that (iv) occurs is $\mathbb{P}(\text{Exp}(\lambda) > z) = e^{-\lambda z}$ and $Y_i \sim \min\{\text{Exp}(\lambda), z\}$ with $\mathbb{E}(Y_i) = \frac{1}{\lambda}(1 - e^{-\lambda z})$, so by taking z to be sufficiently large with respect to n, we can assume that event (iv) never occurs. This means that every time event (i) occurs, ALG pays, on expectation, $1 + 2/\lambda$ for matching r_{2i} and r'_{2i} (and that this match occurs before time t_{2i+1}, i.e., the arrival time of the next requests).

Since every time (i) occurs, ALG pays an additional space cost of 1 for matching (across) r_{2i-1} and r'_{2i-1} and every time (ii) occurs, ALG pays a time cost of 2ϵ, it follows that

$$\begin{aligned}
\mathbb{E}\left(\text{cost}_{ALG}(R)\right) &\ge n(p(2 + 2/\lambda) + (1-p)2\epsilon) \\
&= 2n(p(1 + 1/\lambda) + (1-p)\epsilon) \\
&> 2n(\lambda\epsilon(1 - \lambda\epsilon)(1 + 1/\lambda) + (1 - \lambda\epsilon)\epsilon) \\
&= (1 - \lambda\epsilon)2\epsilon n(\lambda(1 + 1/\lambda) + 1) \\
&= (1 - \lambda\epsilon)2\epsilon n(\lambda + 2)\,.
\end{aligned}$$

By taking $1/\epsilon$ to be sufficiently large with respect to λ (yet, much smaller than n), we conclude that $\mathbb{E}(\text{cost}_{ALG}(R)) > 3\text{cost}_{OPT}(R)$ due to the assumption that $\lambda > 1$.

Acknowledgments

We are indebted to Shay Kutten and Roger Wattenhofer for their help in making this paper happen.

References

1. Aggarwal, G., Goel, G., Karande, C., Mehta, A.: Online Vertex-Weighted Bipartite Matching and Single-bid Budgeted Allocations. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. pp. 1253–1264 (2011)
2. Azar, Y., Chiplunkar, A., Kaplan, H.: Polylogarithmic Bounds on the Competitiveness of Min-cost Perfect Matching with Delays. In: Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. pp. 1051–1061 (2017)
3. Birnbaum, B.E., Mathieu, C.: On-line bipartite matching made simple. SIGACT News 39(1), 80–87 (2008)
4. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press 62(1), 1–20 (1998)
5. Devanur, N.R., Jain, K., Kleinberg, R.D.: Randomized Primal-Dual analysis of RANKING for Online BiPartite Matching. In: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. pp. 101–107 (2013)
6. Edmonds, J.: Maximum matching and a polyhedron with 0, 1-vertices. Journal of Research of the National Bureau of Standards B 69, 125–130 (1965)
7. Edmonds, J.: Paths, trees, and flowers. Canadian Journal of Mathematics 17, 449–467 (1965)
8. Emek, Y., Kutten, S., Wattenhofer, R.: Online Matching: Haste makes Waste! In: 48th Annual Symposium on Theory of Computing (STOC) (Jun 2016), a full version can be obtained from http://ie.technion.ac.il/~yemek/Publications/omhw.pdf
9. Goel, G., Mehta, A.: Online budgeted matching in random input models with applications to Adwords. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA. pp. 982–991 (2008)
10. Kalyanasundaram, B., Pruhs, K.: Online Weighted Matching. J. Algorithms 14(3), 478–488 (1993)
11. Karp, R.M., Vazirani, U.V., Vazirani, V.V.: An Optimal Algorithm for On-line Bipartite Matching. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing. pp. 352–358 (1990)
12. Khuller, S., Mitchell, S.G., Vazirani, V.V.: On-Line Algorithms for Weighted Bipartite Matching and Stable Marriages. Theor. Comput. Sci. 127(2), 255–267 (1994)
13. Mehta, A.: Online Matching and Ad Allocation. Foundations and Trends in Theoretical Computer Science 8(4), 265–368 (2013)
14. Mehta, A., Saberi, A., Vazirani, U.V., Vazirani, V.V.: AdWords and Generalized On-line Matching. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS). pp. 264–273 (2005)
15. Meyerson, A., Nanavati, A., Poplawski, L.J.: Randomized online algorithms for minimum metric bipartite matching. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA (2006)
16. Miyazaki, S.: On the advice complexity of online bipartite matching and online stable marriage. Inf. Process. Lett. 114(12), 714–717 (2014)
17. Naor, J., Wajc, D.: Near-Optimum Online Ad Allocation for Targeted Advertising. In: Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC. pp. 131–148 (2015)
18. Wang, Y., Wattenhofer, R.: Perfect Bipartite Matching with Delays, submitted