# Online Matching with Size-Based and Convex Delays

**Junhao Gan**
School of Computing and Information Systems, The University of Melbourne, Australia

**Xiao Sun**
School of Computing and Information Systems, The University of Melbourne, Australia

**Seeun William Umboh**
School of Computing and Information Systems, The University of Melbourne, Australia
ARC Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Melbourne, Australia

## Abstract

We study the online min-cost perfect matching with delay (MPMD) problem where $m$ requests arrive in a metric space of $n$ points. In MPMD, an algorithm can choose to match a request or to delay, and the objective is to minimise the sum of connection and delay costs. The connection cost of a match is the distance between the locations of two matched requests in the metric, and the increase of the delay cost is a function of the set of unmatched requests at every moment. In this paper, we study two different types of delay functions, size-based (MPMD-Size) and convex delays (MPMD-Convex).

The study of MPMD-Size was initiated by Deryckere and Umboh (APPROX/RANDOM 2023) where the instantaneous delay increment is a non-negative monotone function of the number of unmatched requests. We give an exponential improvement on the lower bounds for the deterministic and randomized competitive ratios. We also give improved upper bounds in terms of $n$, as opposed to Deryckere and Umboh's upper bounds that were functions of $m$, which can be much larger than $n$. Our results settle the deterministic competitive ratio (up to constants). At the heart of these results is a succinct encoding scheme of MPMD-Size on a given $n$-point metric as a metrical task system problem on a $2^{n-1}$-point metric.

We also consider MPMD-Convex proposed by Liu et al. (ISAAC 2018) where the delay cost incurred by each request is a uniform convex delay function of the time difference between its arrival time and the moment that it is matched by the algorithm. They focused on delay functions $f$ that are unbounded, non-decreasing, continuous, and satisfy $f(0) = f'(0) = 0$, and showed that the deterministic competitive ratio is $\Omega(n)$ for $n$-point uniform metrics. We show that, surprisingly, when $f$ is a non-negative, monotone polynomial with $f'(0) > 0$, there is an $O(1)$-competitive deterministic algorithm for uniform metrics. Our result completes our understanding of MPMD-Convex on uniform metrics for a broad class of functions.

**2012 ACM Subject Classification** Theory of computation → Online algorithms

**Keywords and phrases** online algorithms with delays, competitive analysis, matching

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2026.25

**Category** APPROX

**Related Version** *Full Version*: https://arxiv.org/abs/2607.00536

**Funding** *Junhao Gan*: Supported in part by the Australian Government through the Australian Research Council ARC DP230102908.
*Xiao Sun*: Supported by the Australian Government through the Australian Research Council DP240101353 and by the University of Melbourne through the Melbourne Research Scholarship.
*Seeun William Umboh*: Supported by the Australian Government through the Australian Research Council DP240101353.





# 1 Introduction

In recent years, there has been much interest in revisiting classical online problems through the lens of online problems with delay (e.g. [3, 17, 13]). In such problems, instead of serving every request immediately, one can choose to delay the service of a request while accumulating a delay cost, and the total cost of a solution is the cost of serving all requests plus the accumulated delay costs. In such problems, the trade-off between service cost and delay cost is at the center of the decision making of an online algorithm: the algorithm can wait until many requests have arrived and then apply an offline algorithm to serve them cheaply as a batch, but this will incur a large delay cost.

In this paper, we study the Min-Cost Perfect Matching with Delay (MPMD) problem proposed by Emek et al. [16]. In MPMD, there is an underlying metric space of $n$ points in which $m$ requests arrive over time at different points. At every moment when there is an available request, the online algorithm can choose to either match it with another request and incur a *connection cost* equal to the distance between the locations of two requests, or to leave it unmatched and incur a *delay cost*.

MPMD has many intuitive interpretations and real-life applications. For example, suppose that one needs to design a mechanism for an online game platform which always needs to match two players to compete against each other. The mechanism wants to match two players who have similar levels to each other as this could reduce their dissatisfaction, which is a connection cost that the platform wants to minimize. The platform is allowed to choose not to match a player immediately but to wait for another player whose level is close enough to the first player to join the game and then to match them, so that the connection cost can be reduced. However, if one player has been waiting too long, they might lose patience and become more and more unsatisfied with this game platform, which will incur more delay cost instead. Therefore, the online algorithm needs to balance the trade-off between these two types of costs, which is at the center of studies in this area. This kind of problems are inherently online since we do not know what will come in the future, and once we have made a decision, it is irreversible.

**Delay costs.** While the connection cost of a match is the distance between the locations of the two requests that are matched, there are various delay models according to how the delay cost increases at every moment when there is at least one request pending. Some delay models apply a uniform delay function that the total delay of each request is a function of the time different between its arrival and being matched, e.g. linear [16], convex [20], and concave delays [5]. The most general framework is that the delay increment is a function of the set of requests that are pending at every moment and the algorithm is non-clairvoyant that it does not know future delay functions, e.g. set and size-based delays [14].

**Competitive ratios: $n$ vs $m$.** Prior studies on these delay models have achieved results that express competitive ratios as functions of $n$ or $m$. Note that $m$ can be incomparably larger than $n$ when the request sequence is sufficiently long, and such algorithms can perform badly even on a simple one-point metric when $m$ tends to infinity (e.g. [4]). Thus, we aim at designing algorithms whose competitive ratios depend only on $n$.

## 1.1 Our Contributions

In this paper, we make contributions to the study of two generalizations of linear delay functions mentioned above that are much less understood than the others: size-based and convex delays.

**Size-based delay (MPMD-Size).** In the size-based delay setting, at every step the delay increment is a monotone function of the number of currently pending requests (See Section 3.1 for a formal definition). We prove bounds on the deterministic and randomized competitive ratios in terms of $n$. Previously, only upper bounds in terms of $m$ are known.

▶ **Theorem 1.1.** *For every $n$-point metric, the deterministic competitive ratio of MPMD-Size is $\Theta(2^n)$.*

▶ **Theorem 1.2.** *For every $n$-point metric, the randomized competitive ratio (against an oblivious adversary) of MPMD-Size is $O(n^2)$ and $\Omega(n)$.*

We emphasize that the lower bounds in Theorems 1.1 and 1.2 are universal lower bounds: they apply to *every* metric space.

**Table 1** Comparisons between the results from [14] and our new results for MPMD-Size. The lower bounds of [14] only apply to uniform metrics while ours apply to every metric.

| Competitive Ratio | Results from [14] | Our Results |
|---|---|---|
| Deterministic | $O(2^m)$ and $\Omega(n)$ | $\Theta(2^n)$ |
| Randomized | $O(m^4)$ and $\Omega(\log n)$ | $O(n^2)$ and $\Omega(n)$ |

Theorems 1.1 and 1.2 improve upon previous results on size-based delays by Deryckere and Umboh [14] in two ways. First, the previous upper bounds for both deterministic and randomized algorithms are expressed as functions of $m$, the number of requests, while our results are expressed as functions of $n$, the number of points in the given metric space, and thus is an improvement in the more interesting regime where $m \gg n$. Second, we also provide a reduction from MTS to MPMD-Size while theirs is only from MPMD-Size to MTS, which exponentially improves lower bounds for both deterministic and randomized algorithms on *every* $n$-point metric. In contrast, their lower bound only holds for uniform metrics. Our results settle the deterministic competitive ratio, up to a constant. See Table 1 for a detailed comparison between our results and those by Deryckere and Umboh.

**Convex delay (MPMD-Convex).** We also consider the convex delays on $n$-point uniform metrics, first studied by Liu et al. [20]. Convex delay is applicable to a lot of real-life scenarios as well, e.g., a gaming platform where players think it is acceptable to wait for a short period yet a longer delay can be more and more frustrating and even affect the player's willingness to join the platform again. In this problem, there is a non-decreasing delay function $f$ associated with every request and the delay cost of request $r$ incurred until $t$ is $f(t - a(r))$, where $a(r)$ is its arrival time, and the total delay cost is the sum of every request's individual delay. The previous study by Liu et al. focused on the setting where $f$ is non-decreasing, continuous, unbounded and satisfy $f(0) = f'(0) = 0$. They gave a deterministic lower bound of $\Omega(n)$ and a matching upper bound when $f$ is also a monomial.

We consider polynomials $f$ such that $f'(0) > 0$. Prior to our work, the only known result is that when $f$ is linear, the deterministic competitive ratio is $O(1)$ [2]. Our second main result fills this gap and completes our understanding of the landscape of the competitive ratio of MPMD under uniform convex monotone polynomial delays on uniform metrics.

▶ **Theorem 1.3** (Informal; see Theorem 4.1)**.** *For any fixed convex monotone polynomial delay $f$ such that $f'(0) > 0$, there is an $O(1)$-competitive deterministic algorithm for MPMD under delay $f$ on all $n$-point uniform metrics.*

Surprisingly, for different delay functions, their local derivatives at $t = 0$ can totally change the competitiveness, even when they have the same superlinear asymptotic behavior when $t \to \infty$. For example, the deterministic competitive ratio is $\Omega(n)$ for $f(t) = t^2$, yet a minor perturbation to $f(t) = (t + \varepsilon)^2 - \varepsilon^2$ by any small $\varepsilon > 0$ completely changes it to $O(1)$. We will see in the proof that it is due to the approximability between $f(t)$ and the linear delay $g(t) = t$: as long as $f(0) = f'(0) = 0$, the ratio between $f$ and $g$ tends to 0 when $t$ tends to 0 and so we cannot use $g$ to approximate $f$ within positive constant factors. The $O(1)$-competitive result applies to a broader class of functions beyond polynomials as well, even some exponential functions; see Remark 4.18 for further discussion.

## 1.2 Our Techniques

**MPMD-Size.** We employ a similar high-level approach as Deryckere and Umboh [14] of encoding MPMD-Size as a *metrical task system* (MTS) problem (defined in Section 2). Our key contribution is a significantly more efficient encoding.

The main idea is to define the MTS states so that matching requests in the MPMD-Size instance corresponds to a state transition in the MTS instance, and at each timestep, the instantaneous delay increment in the MPMD-Size instance can be realized by a cost vector in the MTS instance. Deryckere and Umboh used the obvious encoding in which there is an MTS state for each even-sized subset of the requests that have arrived so far, and the current MTS state is the set of requests that have been matched so far. While this makes it trivial to encode the instantaneous delay increment as an MTS cost vector – the cost of a state is exactly the instantaneous delay increment of the requests that have arrived but are not matched – the total number of MTS states is the number of all possible even-sized subsets of the requests, which is exponential in $m$. It also complicates matters when applying MTS algorithms as the MTS state space grows as new requests arrive.

Our key insight that enables a more succinct encoding of MPMD-Size as MTS is the following. It never hurts to match two pending requests if they are co-located and so the number of unmatched requests at each point is either 0 or 1. Thus, to determine the number of unmatched requests (which in turn determines the instantaneous delay increment), it suffices to keep track for each point, whether there is an unmatched request at that point. This, in turn, can be done by keeping track of the parity of the number of times a request on that point has been previously matched (see the discussion following Lemma 3.1). Therefore, we can simply express the current state of an MPMD-Size algorithm using an $n$-dimensional Boolean vector. Thus, the total number of MTS states is $O(2^n)$.

Finally, our reductions between MTS and MPMD-Size go in both directions. This lets us improve the lower bounds for both deterministic and randomized algorithms exponentially, and in particular, settle the deterministic competitive ratio (up to constants).

**MPMD-Convex on uniform metrics.** Different from the linear delay, a convex delay function has an unbounded increase rate when it has been pending unmatched for enough long time. As a result, a reasonable algorithm should avoid this case, otherwise even a short period of waiting may lead to huge delay increase. Therefore, we introduce the concept of $T$-impatient algorithms, where a request that has been pending for time $T$ must be matched immediately once there is another available request anywhere in the metric space. In this way, we naturally divide the time axis into two parts according to whether there is a request that has been pending for longer than time $T$ at the moment. If there is no request that has been pending for longer than time $T$, then the delay increase rate of every request is upper and lower bounded by two positive constants (to guarantee the positive lower bound, the property that

$f'(0) > 0$ is necessary) and we can use the linear delay to approximate it up to a constant factor. As for the other case, based on the $T$-impatient property of the algorithm, the only reason that a request has been pending that long is because there is neither another pending request nor the arrival of a new request to be matched with, which means that there must be a request continuously pending throughout this time period under the optimal solution as well, which provides a lower bound of the optimal cost. Luckily, for all monotone polynomials, such a lower bound is within a constant factor of the delay incurred under the algorithm.

Therefore, to design an algorithm for monotone polynomial delays such that $f'(0) > 0$ on uniform metrics, it can be reduced to designing a $T$-impatient algorithm whose competitive ratio is a constant for linear delay on uniform metrics. We show that the seminal algorithm for MPMD on tree metrics by Azar et al. [2] does not satisfy the $T$-impatient property while being implemented on uniform metrics, and thus propose a new algorithm and prove it has the desired property.

## 1.3 Related Works

**Linear delay.** When MPMD was first introduced by Emek et al. [16], the delay function they studied is the uniform linear delay. They proposed a randomized algorithm with competitive ratio $O(\log^2 n + \log \Delta)$, where $n$ is the number of points in the metric space and $\Delta$ is its aspect ratio. Azar et al. [2] improved the upper bound of the randomized competitive ratio for linear delays to $O(\log n)$ by designing an algorithm based on the randomized tree embedding technique into weighted HSTs from [6]. The current best lower bound for the uniform linear delay function is $\Omega(\log n / \log \log n)$ of any randomized algorithm from [1], which shows a gap between known upper and lower bounds even for the most classical delay setting.

The above results that express competitive ratios in terms of $n$ and $\Delta$ usually require the algorithm to know the metric space beforehand, and such competitive ratios are independent of the length of the request sequence. However, there are also results where the competitive ratio is expressed in terms of $m$, the number of requests in the input sequence, while the algorithm does not need to know the metric space prior to its execution: the first such result was proposed by Bienkowski et al. [9] and their deterministic algorithm is $O(m^{2.46})$-competitive. Bienkowski et al. [8] later improved the result to $O(m)$ by an adaptation of the deterministic primal-dual moat growing technique, and independently and concurrently, Azar and Fanani [4] devised a sublinear $O(m^{0.59})$-competitive deterministic algorithm. So far, the best result that expresses the competitive ratio as a function of $m$ is an $O(\log^5 m)$-competitive deterministic algorithm by Dufay and Wattenhofer [15].

**Concave delay.** Azar et al. [5] considered the MPMD problem with uniform concave delays. They designed an $O(1)$-competitive deterministic algorithm for the single point and an $O(\log n)$-competitive randomized algorithm for general $n$-point metric spaces. Deryckere and Umboh [14] generalized the primal-dual framework from [8] and proposed an $O(m)$-competitive deterministic algorithm for uniform concave delays.

**Convex delay.** Liu et al. [20] considered the MPMD problem with uniform convex delays on $n$-point uniform metric spaces. They studied monomial functions $f(t) = t^\alpha$ such that $\alpha > 1$ (where $t$ is the time difference between arrival and being matched for each request). They provided a deterministic $O(n)$-competitive algorithm and showed a matching deterministic lower bound of $\Omega(n)$ for all uniform non-decreasing unbounded continuous delay functions such that $f(0) = f'(0) = 0$. This result demonstrates a gap between the solutions for the cases with linear and convex delays, as the algorithm proposed by Azar et al. [2] achieves

constant competitive ratio on uniform metrics because uniform metrics can be seen as trees of height 1. They also provided a randomized $\Omega(\log n)$ lower bound for delay functions in the same class. There is no result for convex delays beyond uniform metrics yet.

**Set and size-based delays.** In all of the above studies, there is one uniform delay function $f(t)$ known to the algorithm in advance that applies to every request from the input, suggesting how its delay cost increases as time goes by, and the total delay cost is the sum of delay costs incurred by each request. Deryckere and Umboh [14] studied another model of delay function that is not accumulated request-wise but step-wise: at every time step $t$, the instantaneous delay cost is incurred as a function of the set of unmatched requests during this step. This set-delay model studies MPMD problem in the least restrictive way, as there is essentially no restriction on the structure of the delay functions and the setting is non-clairvoyant, i.e. the algorithm does not know future delays. Similar non-clairvoyant delay settings have been studied in other online problems with delays, e.g., TCP Acknowledgment Problem [7] and Subadditive Joint Replenishment Problem [19, 18]. Unfortunately, under the most general set-delay setting, every deterministic algorithm has competitive ratio of at least $\Omega(\Delta)$, where $\Delta$ is the aspect ratio of the metric space. Instead, Deryckere and Umboh further restricted to the case of size-based delays, which have natural applications in practical settings with service-level agreements such as cloud computing. They showed that there is an $O(2^m)$-competitive deterministic algorithm and an $O(m^4)$-competitive randomized algorithm by a reduction to the classical MTS problem introduced by Borodin et al. [10] and applying the best upper bounds for both deterministic and randomized algorithms by Borodin et al. and Bubeck et al. [12] respectively. They also provided lower bounds in terms of $n$, which is $\Omega(n)$ for deterministic algorithms and $\Omega(\log n)$ for randomized algorithms. These upper and lower bounds show significant gaps between $m$ and $n$ in the scenarios where a request sequence of unbounded length is generated on a given finite metric.

**Paper organization.** We start by introducing the formal definition of MTS in Section 2. In Section 3, we show the reductions between MPMD-Size and MTS on both directions and hence complete the proof of Theorems 1.1 and 1.2. We study MPMD-Convex on uniform metrics and prove Theorem 1.3 in Section 4. We conclude by some discussion on the results as well as open problems.

## 2 Preliminaries

In this section, we introduce the definition and properties of the Metrical Task System problem, which will be used in Section 3 to prove the main results of our paper on MPMD-Size. The formal definitions and notations of MPMD-Size and MPMD-Convex, two different delay models of MPMD that we study, are in Sections 3 and 4 respectively.

The metrical task system (MTS) problem, introduced by Borodin et al. [10], can be defined as a sequence of requests $\{\rho_t\}_{t\geq 1}$ on a given finite metric space $(V, d)$ and an initial state $s_0 \in V$ of the server. Each request $\rho_t : V \to \mathbb{R}_{\geq 0}$ is a non-negative cost function on the state space $V$, illustrating the cost of the server to process request $\rho_t$ in different points from $V$. Upon the arrival of request $\rho_t$ at every step $t \geq 1$, the server can be moved to some state $s_t \in V$ to process $\rho_t$, and this incurs *service cost* of $\rho_t(s_t)$. Since MTS is an online problem, the state $s_t$ at every step $t$ must be output without knowing future requests. The total cost of a solution $\{s_t\}_{t\geq 1}$ that expresses the state $s_t$ where the server processes request

$\rho_t$ at every step $t \geq 1$ consists of service cost and *transition cost*, where the transition cost at every step $t$ is the distance $d(s_{t-1}, s_t)$ of moving to $s_t$ from its previous state $s_{t-1}$. As a result, the cost of a solution $\{s_t\}_{t\geq 1}$ is defined as:

$$\sum_{t\geq 1}[\rho_t(s_t) + d(s_{t-1}, s_t)].$$

For MTS problems, we say that an online algorithm $\mathcal{A}$ is *lazy* if: at every step $t$ when it moves from $s_{t-1}$ to a different state $s_t$ to process $\rho_t$, it must hold that $\rho_t(s_{t-1}) > 0$. Intuitively speaking, a lazy algorithm $\mathcal{A}$ only makes a move when its current state incurs positive cost for the upcoming request while always staying still if the service cost of the new request in the current state is zero.

▶ **Lemma 2.1.** *Given any online algorithm $\mathcal{A}$ for MTS, there is an online mechanism that transforms $\mathcal{A}$ to a lazy algorithm without increasing the cost.*

From now on, we can assume that every MTS algorithm that we use is lazy.

Let MTS-Single denote the subclass of MTS problem where every request $\rho_t$ incurs positive cost at exactly one point $v_t \in V$, i.e. $\rho_t(v_t) > 0$ and $\rho_t(v) = 0$ for every $v \in V \setminus \{v_t\}$. When a metric space is given, we use $c_{\mathrm{m}}$ to denote the minimum possible worst-case competitive ratio that can be achieved by an algorithm for MTS problem on it, and $c_{\mathrm{ms}}$ to denote that can be achieved for MTS-Single problem. The following result is folklore for both deterministic and randomized algorithms:

▶ **Lemma 2.2.** $c_{\mathrm{m}} \leq c_{\mathrm{ms}}$.

The proofs of Lemmas 2.1 and 2.2 can be found in the full version. They will be used in Section 3 to establish the reductions between MPMD-Size and MTS.

# 3 Reduction between MPMD-Size and MTS

In this section, we first formally define the MPMD problem with size-based delays (MPMD-Size), and then describe the reduction between it and MTS.

## 3.1 Definition and Property of MPMD-Size

The MPMD problem is defined on a metric space $(V, d)$, where $V$ is the set of points and $d : V \times V \to \mathbb{R}_{\geq 0}$ is the distance function. An online input consists of a sequence of requests that arrive over time. Every request $r$ is defined by two characteristics: its arrival time $a(r)$ and location $\ell(r) \in V$. We view the problem in the discrete way that any request can only arrive at a time step $t \in \mathbb{N}_{>0}$, yet it is possible that multiple requests arrive at the same time step $t$. This model captures the original definition by Emek et al. [16] when we divide the continuous time axis into discrete steps where a step covers an interval of infinitesimal length.

Suppose $Q_t$ is the set of requests that have arrived up to and including time step $t$. At every step, the instantaneous delay function $f_t : 2^{Q_t} \to \mathbb{R}_{\geq 0}$ specifies the increment of delay cost at this step for every possible $P_t \subseteq Q_t$, where $P_t$ is the set of requests that are pending during time step $t$. At every step $t$, the algorithm first sees the newly-arriving requests (if any) and the instantaneous delay function $f_t$, then makes a decision on matching some requests. A match between two requests $r_1$ and $r_2$ incurs a connection cost of $d(\ell(r_1), \ell(r_2))$. This process results in an updated $P_t$, and the instantaneous delay $f_t(P_t)$ is then incurred

after the matching performed at step $t$. Observe that the definition is consistent with metrical task systems (defined in Section 2) in the sense that at every step, the algorithm first sees the new input, then decides which state to move to (possibly the same state), and incurs the cost of the resulting state. The goal is to minimize the sum of its connection cost and delay cost incurred among all steps.

In the MPMD-Size problem, the instantaneous delay function incurred at time step $t$ can be expressed as $f_t(|P_t|)$, where $f_t : \mathbb{N}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a monotone function. We note that the well-studied linear delay cost is a special case of the size-based delay where $f_t(|P_t|) = |P_t|$ at every moment $t$.

We say that an online algorithm $\mathcal{A}$ is *smart* if no two requests are pending at the same point during any time step. One important property of sized-based delay functions is that we can without loss of generality assume that every algorithm or solution is smart, because of the following lemma whose proof can be found in the full version:

▶ **Lemma 3.1.** *For size-based delays, given any online algorithm $\mathcal{A}$, there is an online mechanism that transforms $\mathcal{A}$ to a smart algorithm without increasing the cost.*

Lemma 3.1 provides us with not only a nice property of the optimal solution, but also a mechanism of online transformations that allows us to always automatically match two requests at the same point immediately at any moment. As a result, we do not need to explicitly describe the a match of two requests at the same point in the metric anymore. Based on this, we can assume that at every point $v \in V$ there is either one or zero pending request at every time step $t$, and there are at most $n$ requests pending in the whole metric space $V$. Moreover, at every given time step, we can refer to a match between two requests at two different points as a match between these two points without ambiguity. Throughout the paper, we may use expressions such as "the algorithm matches point $v_1$ with point $v_2$" and "point $v$ has been matched for an odd number of times", where the former refers to a match between requests at these two points $v_1$ and $v_2$, and the latter refers to the total number of times where a request at $v$ has been matched with some request at another point. As we have discussed in Section 1.2, this allows us to encode the state of an algorithm with a Boolean vector that has an entry over every point in $V$, which builds the foundation of our reductions.

## 3.2 Construction of MTS Metric Space

Before we formally describe the reduction between two problems, we first specify the corresponding metric space $(\mathcal{K}^V, D)$ on which the request sequence for both MTS and MTS-Single problems is based, given a metric space $(V, d)$ for an MPMD-Size instance.

▶ **Definition 3.2.** *Given any finite set $V$, define $\mathcal{K}^V := \{\mathbf{u} \in \{0,1\}^V \mid \sum_{v \in V} \mathbf{u}(v) = 0 \mod 2\}$, where $\mathbf{u}(v)$ is the entry of $\mathbf{u}$ on a given $v \in V$ for every $\mathbf{u} \in \{0,1\}^V$.*

Observe that $|\mathcal{K}^V| = 2^{n-1}$.

**Intuition for $(\mathcal{K}^V, D)$.** To have an intuitive understanding of the structure of $(\mathcal{K}^V, D)$ defined on a metric $(V, d)$ and its addition operation, we can give a Boolean variable to every point in $V$, and see every match between two points in $V$ as an operation that flips the Boolean variable associated with these two points simultaneously. If the default setting is that the Boolean variable associated with every point in $V$ is 0, then $\mathcal{K}^V$ is exactly the set of states that can be attained by applying the matching operations, where each state is a

Boolean vector that specifies the value of the Boolean variable associated with every point in $V$. With this intuition, if the cost of a match is exactly the distance of the two points involved in it, then the distance between two states in $\mathcal{K}^V$ should be the minimum cost to transform one state into another by applying matching operations.

As a result, we define the concept of the induced matching of every element in $\mathcal{K}^V$ (Definition 3.5) and use it to express the distance $D$ on $\mathcal{K}^V$. We will see that this definition of distance indeed satisfies the min-cost transformation property between every two states (Lemma 3.18).

▶ **Definition 3.3** (Addition operation on $\{0,1\}^V$). *Let $\oplus$ denote the XOR Boolean operation over $\{0,1\}$, i.e. $a \oplus b = 0$ if $a = b$ and $a \oplus b = 1$ if $a \neq b$, for $a, b \in \{0,1\}$. For every $\mathbf{u}_1, \mathbf{u}_2 \in \{0,1\}^V$, $\mathbf{u}_1 \oplus \mathbf{u}_2 := (\mathbf{u}_1(v) \oplus \mathbf{u}_2(v))_{v \in V}$.*

We use $\mathbf{0}$ to denote $(0)_{v \in V}$ and $\mathbf{1}$ to denote $(1)_{v \in V}$. We first note that $\mathcal{K}^V$ is a closed subgroup of $\{0,1\}^V$ under this addition operation.

▷ Claim 3.4. For every $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{K}^V$, we have $\mathbf{u}_1 \oplus \mathbf{u}_2 \in \mathcal{K}^V$.

Proof. If $\sum_{v \in V} \mathbf{u}_1(v) = \sum_{v \in V} \mathbf{u}_2(v) = 0 \mod 2$, then $\sum_{v \in V} (\mathbf{u}_1 \oplus \mathbf{u}_2)(v) = \sum_{v \in V} (\mathbf{u}_1(v) \oplus \mathbf{u}_2(v)) = \sum_{v \in V} \mathbf{u}_1(v) + \sum_{v \in V} \mathbf{u}_2(v) = 0 \mod 2$. ◁

▶ **Definition 3.5** (Induced matching of an element in $\mathcal{K}^V$). *Given $\mathbf{u} \in \mathcal{K}^V$, let $T_{\mathbf{u}} = \{v \in V \mid \mathbf{u}(v) = 1\}$. Define the* induced matching *of $\mathbf{u}$ to be an arbitrary min-cost perfect matching of points in $T_{\mathbf{u}}$ in the metric space $(V, d)$.*[1] *For every min-cost perfect matching $M$ of $T_{\mathbf{u}}$, we associate the vector $\mathbf{u}$ with $M$.*

▶ **Definition 3.6** (Distance $D$ on $\mathcal{K}^V$). *Given $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{K}^V$, define their distance $D(\mathbf{u}_1, \mathbf{u}_2)$ to be the weight of a min-cost perfect matching induced by $\mathbf{u}_1 \oplus \mathbf{u}_2$.*

See the full version for the proof of the following claim.

▷ Claim 3.7. $(\mathcal{K}^V, D)$ is a metric space.

## 3.3 From MPMD-Size to MTS

Given any instance of MPMD-Size on the metric space $(V, d)$, we first describe the reduction from it to an instance of MTS on the corresponding $(\mathcal{K}^V, D)$, as well as how to translate an online algorithm for MTS to one for the original MPMD-Size.

We introduce some notations to express the state of an MPMD-Size instance under the execution of a given algorithm at step $t$, including: the total arrivals of requests at different points, and the effect of all matches made by the algorithm so far.

▶ **Definition 3.8.** *Given an algorithm for MPMD-Size on the metric space $(V, d)$, for every step $t \geq 1$, let $R_t \in \{0,1\}^V$ be the Boolean vector such that $R_t(v) = 1$ if and only if the parity of the number of requests that have arrived at $v \in V$ until (including) step $t$ is odd, $M_t \in \mathcal{K}^V$ be the Boolean vector such that $M_t(v) = 1$ if and only if $v$ has been matched for an odd number of times until (including) step $t$. The initial setting ($t = 0$) is that $R_0 = M_0 = \mathbf{0}$.*

In MPMD-Size, at every step $t$, new requests arrive and the instantaneous delay function $f_t$ is revealed at the same time, and an online algorithm needs to make decisions to match according to this information. At each step $t$, we need to handle two situations: when new

[1] A perfect matching exists since $|T_{\mathbf{u}}|$ is even, by the definition of $\mathcal{K}^V$.

requests have arrived at $t$ and before the algorithm has made new decisions yet, and after the algorithm has made decisions at time $t$. The former determines the matchings that the algorithm can make, and the latter determines the instantaneous cost incurred at the end of step $t$. With notations introduced in Definition 3.8, it is easy to see that the former state can be expressed as $R_t \oplus M_{t-1}$ and the latter state can be expressed as $R_t \oplus M_t$. With the smart algorithm property and the mechanism shown in Lemma 3.1, it is not hard to see that we can assume that requests at the same point cancel out each other automatically for state $R_t \oplus M_{t-1}$ as well. We summarize the above discussion in the following observation that expresses the number of request at every point with a Boolean vector:

▶ **Observation 3.9.** *Upon the arrivals of new requests at time step $t$ (before new matches are made by the algorithm at step $t$), the number of pending requests at every point $v$ is equal to $M_{t-1}(v) \oplus R_t(v)$; once the algorithm has made decisions at step $t$, the number of pending requests at $v$ is equal to $M_t(v) \oplus R_t(v)$ when the delay cost is incurred at step $t$.*

If $(M_{t-1} \oplus R_t)(v) = 0$, then there is no request pending at $v$ after new arrivals at $t$, and of course there cannot be any request at $v$ after new matches are performed at $t$ as well, thus leading to the following observation:

▶ **Observation 3.10.** $(R_t \oplus M_{t-1})(v) \leq (R_t \oplus M_t)(v)$ *for every $v \in V$ and time step $t$.*

For every $\mathbf{u} \in \{0,1\}^V$, we use $|\mathbf{u}|$ to denote the number of 1's among all entries of $\mathbf{u}$, i.e. $|\mathbf{u}| = \sum_{v \in V} \mathbf{u}(v)$. During every step $t$ when the delay cost is incurred, $R_t \oplus M_t$ is the Boolean vector that denotes the state of the input under the algorithm as per Observation 3.9. Based on these notations and the property of size-based delays, the delay cost incurred during step $t$ can be expressed as $f_t(|R_t \oplus M_t|)$.

**Construction of MTS instance.** Now we are ready to define the MTS instance that we reduce MPMD-Size to. We set the server's initial position of the MTS instance to be $C_0 = \mathbf{0}$. Upon every time step $t \geq 1$, we assign a request vector $(\rho_t(\mathbf{u}))_{\mathbf{u} \in \mathcal{K}^V}$ to the MTS instance according to $R_t$, the Boolean vector that we have defined to denote the parity of the number of requests that have arrived at each point in $V$ until (and including) $t$, and $f_t$, the instantaneous delay cost function of the MPMD-Size instance incurred at step $t$. More specifically, the cost to process $\rho_t$ at $\mathbf{u}$ is $\rho_t(\mathbf{u}) = f_t(|R_t \oplus \mathbf{u}|)$ for every $\mathbf{u} \in \mathcal{K}^V$.

Let $\mathrm{OPT_m}$ denote the value of the optimal solution of the MTS instance and $\mathrm{OPT_{size}}$ denote that of the MPMD-Size instance.

▶ **Lemma 3.11.** $\mathrm{OPT_{size}} \geq \mathrm{OPT_m}$.

**Proof.** Given any solution $\mathrm{SOL_{size}}$ of MPMD-Size, we convert it to a solution $\mathrm{SOL_m}$ of MTS without increasing the cost. Upon every step $t$, let $M_t$ be the sum of Boolean vectors of matches made by $\mathrm{SOL_{size}}$ so far, then $M_t$ is an element of $\mathcal{K}^V$, and $\mathrm{SOL_m}$ moves from state $M_{t-1}$ to $M_t$ (this is feasible at step 1 since at the beginning $M_0 = \mathbf{0}$ and the initial configuration of the MTS instance is given as $C_0 = \mathbf{0}$ as well).

We now compare the cost of two solutions. At every step $t$, the number of pending requests in the MPMD-Size solution is $|R_t \oplus M_t|$. As a result, the delay cost incurred during step $t$ is $f_t(|R_t \oplus M_t|)$, which is equal to $\rho_t(M_t)$, the cost to process the task $\rho_t$ at $M_t$, based on our setting of $(\rho_t(\mathbf{u}))_{\mathbf{u} \in \mathcal{K}^V}$. By Lemma 3.18, the connection cost of $\mathrm{SOL_{size}}$ at step $t$ is at least the distance $D(M_{t-1}, M_t)$, which equals the cost of MTS to move from $M_{t-1}$ to $M_t$. ◀

**Translating MTS algorithm to MPMD-Size.** We now describe how to transform an algorithm $\mathcal{A}_{\text{m}}$ for the MTS instance to one $\mathcal{A}_{\text{size}}$ for the MPMD-Size instance in an online way. As we see from the proof of Lemma 3.11, there is naturally a one-to-one correspondence between the current MTS state and the current total matching Boolean vector $M_t$ of MPMD-Size, which can be seen as the state of an MPMD-Size algorithm. Therefore, given an online algorithm $\mathcal{A}_{\text{m}}$ for MTS problem on $(\mathcal{K}^V, D)$ that processes request $\rho_t$ at state $C_t$, intuitively, we want to transit MPMD-Size algorithm to state $M_t = C_t$ as well. However, such a transition is not always feasible, as there is no guarantee that there is an available request at every point that the algorithm needs to match to transit to $C_t$. If this happens, then we want to update $M_{t-1}$ to $M_t$ that is the closest possible to $C_t$ among all feasible matchings that can be made: we check every pair of match in the min-cost perfect matching induced by $M_{t-1} \oplus C_t$, and match every feasible pair of requests. See the description of Algorithm 1 for more details.

**Algorithm 1** The translation algorithm of an online algorithm $\mathcal{A}_{\text{m}}$ for MTS on $(\mathcal{K}^V, D)$ to $\mathcal{A}_{\text{size}}$ for MPMD-Size on $(V, d)$.

**Initialize:** $M_0 = \mathbf{0}$ for MPMD-Size, and the initial configuration of MTS is $C_0 = \mathbf{0}$.

**At every step $t$:**

Given $R_t$, the arrivals of requests so far in the MPMD-Size instance and $f_t$ the instantaneous size-based delay cost, task vector $\rho_t$ of MTS is constructed such that $\rho_t(\mathbf{u}) = f_t(|R_t \oplus \mathbf{u}|)$ for every $\mathbf{u} \in \mathcal{K}^V$, and we are given $\mathcal{A}_{\text{m}}$'s state $C_t$ at step $t$.

Upon arrivals of new requests at $t$, the Boolean vector of current available requests in $V$ is $R_t \oplus M_{t-1}$ (Observation 3.9).

Calculate $M_{t-1} \oplus C_t$, the Boolean vector of the matching that $\mathcal{A}_{\text{size}}$ intends to make to transit to state $M_t$. For every edge $(u, v)$ in the min-cost perfect matching induced by $M_{t-1} \oplus C_t$, if $(M_{t-1} \oplus R_t)(u) = (M_{t-1} \oplus R_t)(v) = 1$, then there is a pending request at both $u$ and $v$ after new arrivals at step $t$, and we match $u$ and $v$ in this case.

After all matches have been performed, the new vector $M_t$ is updated accordingly.

▷ Claim 3.12. For every $v \in V$, if $M_t(v) \neq C_t(v)$, then $(M_{t-1} \oplus C_t)(v) = 1$.

Proof. We prove the contrapositive. If $(M_{t-1} \oplus C_t)(v) = 0$, then $\mathcal{A}_{\text{m}}$ does not make any match that involves $v$ at step $t$, and so $M_t(v) = M_{t-1}(v) = C_t(v) = 0$. ◁

The following claim guarantees the local optimality of $\mathcal{A}_{\text{m}}$ as per the transition from $M_{t-1}$ to $M_t$ by matches at step $t$:

▷ Claim 3.13. The connection cost of $\mathcal{A}_{\text{m}}$ at step $t$ is equal to $D(M_{t-1}, M_t)$.

Proof. Let $J_1$ be the set of edges matched by $\mathcal{A}_{\text{m}}$ at step $t$ and $J_2$ be the set of edges in the min-cost perfect matching induced by $M_{t-1} \oplus C_t$, then $J_1 \subseteq J_2$ by the mechanism of $\mathcal{A}_{\text{m}}$. If $J_1'$ is a another perfect matching on $T_{M_{t-1} \oplus M_t} = \{v \in V \mid (M_{t-1} \oplus M_t)(v) = 1\}$ such that the cost of $J_1'$ is smaller than $J_1$, then $(J_2 \setminus J_1) \cup J_1'$ is a perfect matching on $T_{M_{t-1} \oplus C_t} = \{v \in V \mid (M_{t-1} \oplus C_t)(v) = 1\}$ whose cost is smaller than $J_2$, a contradiction to the assumption that $J_2$ is a min-cost perfect matching! ◁

We prove that the cost of $\mathcal{A}_{\text{size}}$ is no more than that of $\mathcal{A}_{\text{m}}$ by the following lemmas. First, we compare the instantaneous delay cost of $\mathcal{A}_{\text{size}}$ incurred at step $t$ with the cost of processing task $\rho_t$ by $\mathcal{A}_{\text{m}}$, and this is where the size-based property of the delay function $f_t$ plays a vital role.

▶ **Lemma 3.14.** *The delay cost at step $t$ by $\mathcal{A}_{\text{size}}$ is no more than the service cost to process request $\rho_t$ by $\mathcal{A}_{\text{m}}$, given that $f_t$ is a size-based delay.*

**Proof.** The delay cost of $\mathcal{A}_{\text{size}}$ at step $t$ is $f_t(|R_t \oplus M_t|)$ and the service cost to process request $\rho_t$ by $\mathcal{A}_{\text{m}}$ is $f_t(|R_t \oplus C_t|)$. By the monotonicity of the size-based delay function, it suffices to show that $|R_t \oplus M_t| \leq |R_t \oplus C_t|$. We do this by constructing an injective map $h_t : P'_t \setminus P_t \to P_t \setminus P'_t$, where

$$P'_t = \{v \in V \mid (R_t \oplus M_t)(v) = 1\}; \qquad P_t = \{v \in V \mid (R_t \oplus C_t)(v) = 1\}.$$

For every $v \in P'_t \setminus P_t$, it holds that $M_t(v) \neq C_t(v)$ according to the definition of $P'_t$ and $P_t$. By Claim 3.12, it must be that $(M_{t-1} \oplus C_t)(v) = 1$. Let $u$ be the request that is matched with $u$ in the min-cost perfect matching induced by $M_{t-1} \oplus C_t$. We now prove that $u \in P_t \setminus P'_t$, and hence we can define $h_t(v) = u$.

By Observation 3.10, $(R_t \oplus M_{t-1})(v) = 1$ for every $v \in P'_t$. By the construction of $\mathcal{A}_{\text{size}}$, if $(R_t \oplus M_{t-1})(v) = 1$ and $M_t(v) \neq C_t(v)$ both hold, then it must be that $(R_t \oplus M_{t-1})(u) = 0$ making $v$ unmatched at step $t$. In this case, it holds that $(R_t \oplus M_t)(u) \leq (R_t \oplus M_{t-1})(u) = 0$ according to Observation 3.10, which implies that $u \notin P'_t$ by definition. On the other hand, $(M_{t-1} \oplus C_t)(u) = 1$ yet $u$ is unmatched at step $t$, which implies that $(R_t \oplus C_t)(u) = (R_t \oplus M_{t-1})(u) \oplus 1 = 1$, and so $u \in P_t$. As a result, $u \in P_t \setminus P'_t$ and $h_t$ is well-defined on $P'_t \setminus P_t$.

We now prove that $h_t$ is injective. For any $v_1 \neq v_2$ from $P'_t \setminus P_t$, $h_t(v_1)$ is matched with $v_1$ and $h_t(v_2)$ is matched with $v_2$ in the min-cost perfect matching induced by $M_{t-1} \oplus C_t$. As a result, $h_t(v_1) \neq h_t(v_2)$ by the matching property, and $h$ is an injection. ◀

The next lemma compares the connection cost of $\mathcal{A}_{\text{size}}$ (which is $D(M_{t-1}, M_t)$ at every step $t$ according to Claim 3.13) with the transition cost of $\mathcal{A}_{\text{m}}$.

▶ **Lemma 3.15.** *$\sum_{t=1}^{T} D(M_{t-1}, M_t) \leq \sum_{t=1}^{T} D(C_{t-1}, C_t)$, where $T$ is the final step of both of the corresponding MTS and MPMD-Size instances.*

**Proof.** We define a potential function $\phi_t = D(M_t, C_t)$ for each step $t$. It is easy to see that $\phi_0 = 0$ and $\phi_t \geq 0$ for every $t$. We prove the telescopic inequalities

$$D(M_{t-1}, M_t) \leq D(C_{t-1}, C_t) - (\phi_t - \phi_{t-1})$$

for every step $t \geq 1$ to draw the conclusion.

First, by the triangle inequality on the metric space $(\mathcal{K}^V, D)$, it holds that

$$D(M_{t-1}, C_t) \leq D(M_{t-1}, C_{t-1}) + D(C_{t-1}, C_t) = D(C_{t-1}, C_t) + \phi_{t-1}.$$

We now show that $D(M_{t-1}, M_t) + \phi_t = D(M_{t-1}, M_t) + D(C_t, M_t) \leq D(M_{t-1}, C_t)$, then we get the desired inequality by summing up these two.

Define

$$T_1 = \{v \in V \mid (M_{t-1} \oplus M_t)(v) = 1\}; \qquad T_2 = \{v \in V \mid (M_t \oplus C_t)(v) = 1\}.$$

We first observe that $T_{M_{t-1} \oplus C_t} = \{v \in V \mid (M_{t-1} \oplus C_t)(v) = 1\} = T_1 \Delta T_2$, according to the definition of $\oplus$ operation of Boolean vectors. By Claim 3.12, for every $v \in T_2$, it must hold that $v \in T_{M_{t-1} \oplus C_t}$ and hence $v \notin T_1$, which implies that $T_1 \cap T_2 = \varnothing$. As a result, we can express $T_{M_{t-1} \oplus C_t} = T_1 \sqcup T_2$[2] . Moreover, for each pair $(u, v)$ in the min-cost perfect

[2] $T_1 \sqcup T_2$ is the disjoint union of $T_1$ and $T_2$.

matching induced by $M_{t-1} \oplus C_t$, either $u, v$ are both in $T_1$, or $u, v$ are both in $T_2$, according to the mechanism of $\mathcal{A}_{\text{size}}$ to update $M_t$ at every step. Therefore, we can divide all matches in the min-cost perfect matching induced by $M_{t-1} \oplus C_t$ into two parts: $J_1$ that forms a matching on $T_1$, and $J_2$ that forms a matching on $T_2$. Using $w(\cdot)$ to denote the weight of a matching, we have $w(J_1) + w(J_2) = D(M_{t-1}, C_t)$ since $J_1 \cup J_2$ forms the min-cost perfect matching induced by $M_{t-1} \oplus C_t$. Moreover, $D(M_{t-1}, M_t) \leq \omega(J_1)$ and $D(M_t, C_t) \leq \omega(J_2)$ by the definition of distance $D$. This proves the desired inequality. ◀

Combining Lemmas 3.14 and 3.15, we conclude that the total cost of $\mathcal{A}_{\text{size}}$ on the MPMD-Size instance on $(V, d)$ is no more than the total cost of $\mathcal{A}_{\text{m}}$ on the MTS instance on $(\mathcal{K}^V, D)$, yielding the following theorem of the reduction.

▶ **Theorem 3.16.** *Let $(V, d)$ be a metric, $c_{\text{m}}$ be the competitive ratio of MTS on $(\mathcal{K}^V, D)$ and $c_{\text{size}}$ be that of MPMD-Size on $(V, d)$. Then, $c_{\text{size}} \leq c_{\text{m}}$.*

Applying the upper bound of $O(N)$ for deterministic algorithms from [10] and that of $O(\log^2 N)$ for randomized algorithms from [12] of MTS on every $N$-point metric gives the upper bounds in Theorems 1.1 and 1.2.

▶ **Corollary 3.17.** *There is an $O(2^n)$-competitive deterministic algorithm and an $O(n^2)$-competitive randomized algorithm for MPMD-Size on every $n$-point metric.*

## 3.4 From MTS-Single to MPMD-Size

In this part, we establish the reduction from any given instance of MTS-Single on $(\mathcal{K}^V, D)$ to one of MPMD-Size on $(V, d)$, where $V$ is a given metric space and $|V| = n$.

**MPMD-Size delay function.** We first state the structure of the size-based delay function $f_t$ of the MPMD-Size instance for every step $t \geq 1$:

$$f_t(N) = \begin{cases} 0 & \text{if } N < n; \\ \varepsilon_t & \text{if } N \geq n. \end{cases}$$

Here $\varepsilon_t$ denotes a positive amount in our construction that will depend on the request of the MTS-Single instance at step $t$. More specifically, at every step $t$ there is exactly one point that is "hit" with a positive cost by the request in MTS-Single, and we use $\varepsilon_t$ to denote the amount of this positive cost.

**Requests of MPMD-Size instance.** We assume that the initial setting of the MTS server is $C_0 = \mathbf{0}$, otherwise we set $R_t \oplus \mathbf{u}_t = C_0 \oplus \mathbf{1}$ and show that the property maintained in the proof of Lemma 3.19 is $M_t \oplus C_t = C_0$ for every step $t$ instead (it is intuitively like "shifting" both $R_t$ and $M_t \oplus C_t$ by $C_0$). At every step $t$ when the new request $\rho_t$ arrives in the MTS-Single instance on $(\mathcal{K}^V, D)$, suppose that $\rho_t(\mathbf{u}_t) = \varepsilon_t > 0$ and $\rho_t(\mathbf{u}) = 0$ for every $\mathbf{u} \in \mathcal{K}^V \setminus \{\mathbf{u}_t\}$, then we set $R_t = \mathbf{u}_t \oplus \mathbf{1}$ for the MPMD-Size instance. To have a more intuitive understanding of how new requests arrive in the metric space at step $t$, we use the difference between $R_{t-1}$ and $R_t$, i.e. $R_{t-1} \oplus R_t$, to denote the new arrivals at step $t$, such that $(R_{t-1} \oplus R_t)(v) = 1$ if and only if a new request arrives at step $t$ at point $v$, for every $v \in V$. Moreover, we can express this amount as $R_{t-1} \oplus R_t = R_{t-1} \oplus \mathbf{u}_t \oplus \mathbf{1}$, which can be computed from $R_{t-1}$ that has been well-defined from previous steps, and $\mathbf{u}_t$ defined by the new request of MTS-Single instance at step $t$. As a result, the MPMD-Size instance that we reduce MTS-Single to is well-defined.

The following lemma provides a lower bound of a solution's connection cost at step $t$, given $M_{t-1}$ and $M_t$, the two consecutive states of matches that have been made so far.

▶ **Lemma 3.18.** *The cost of matches made at step $t$ is at least $D(M_{t-1}, M_t)$.*

**Proof.** Let $M(v)$ denote the number of times that the algorithm matches $v$ with another point at step $t$, then $M(v)$ is either 0 or 1 based on the assumption that there is at most one available request at every $v \in V$ and so we naturally regard $M$ as a Boolean vector. For every $v \in V$, it holds that $M_t(v) = M_{t-1}(v) \oplus M(v)$, according to the definition of $M_t$ and $M_{t-1}$ and the addition operation on Boolean vectors. As a result, $M$ is a perfect matching of $T_{M_{t-1} \oplus M_t} = \{v \in V \mid (M_{t-1} \oplus M_t)(v) = 1\}$ (since we only consider matches between different points). The statement follows by Definition 3.6. ◀

Let $\mathrm{OPT}_{\mathrm{ms}}$ denote the value of the optimal solution of the MTS-Single instance and $\mathrm{OPT}_{\mathrm{size}}$ denote the that of the MPMD-Size instance.

▶ **Lemma 3.19.** $\mathrm{OPT}_{\mathrm{ms}} \geq \mathrm{OPT}_{\mathrm{size}}$.

See the full version of our paper for the proof of Lemma 3.19.

The next step is to convert an online algorithm $\mathcal{A}_{\mathrm{size}}$ for the MPMD-Size instance to one $\mathcal{A}_{\mathrm{ms}}$ for the MTS-Single instance without increasing the cost. At every step $t$ after new matches have been made by $\mathcal{A}_{\mathrm{size}}$ and $M_t$ is updated, $\mathcal{A}_{\mathrm{ms}}$ processes request $\rho_t$ in state $M_t$.

▶ **Lemma 3.20.** *The cost of $\mathcal{A}_{\mathrm{ms}}$ is no more than that of $\mathcal{A}_{\mathrm{size}}$.*

**Proof.** It is easy to see the transition cost of $\mathcal{A}_{\mathrm{ms}}$ is no more than the connection cost of $\mathcal{A}_{\mathrm{size}}$. This is because the matching made by $\mathcal{A}_{\mathrm{size}}$ at $t$ must be at least the value of a min-cost perfect matching induced by $M_t \oplus M_{t-1}$ according to Lemma 3.18, which is equal to $D(M_{t-1}, M_t)$, the cost of moving from the previous state $M_{t-1}$ to a new state $M_t$ at step $t$ in the MTS-Single instance.

On the other hand, the delay cost incurred for MPMD-Size during step $t$ is equal to $\varepsilon_t = \rho_t(\mathbf{u}_t)$ (recall that $\mathbf{u}_t$ is the only state in $\mathcal{K}^V$ where $\rho_t$ incurs positive cost) if and only if $R_t \oplus M_t = \mathbf{1}$, which is equivalent to that $M_t = \mathbf{u}_t$ based on our setting of the MPMD-Size instance that $R_t \oplus \mathbf{u}_t = \mathbf{1}$. Thus, the service cost of $\mathcal{A}_{\mathrm{ms}}$ is equal to the delay cost of $\mathcal{A}_{\mathrm{size}}$. ◀

Now the reduction from MTS-Single to MPMD-Size has been established as the following:

▶ **Theorem 3.21.** *Let $(V, d)$ be a metric, $c_{\mathrm{ms}}$ be the competitive ratio of MTS-Single on $(\mathcal{K}^V, D)$ and $c_{\mathrm{size}}$ be that of MPMD-Size on $(V, d)$. Then, $c_{\mathrm{ms}} \leq c_{\mathrm{size}}$.*

Combined with Lemma 2.2, we can use $c_{\mathrm{m}}$, the competitive ratio of the general MTS problem, to provide lower bounds for the MPMD-Size problem. Therefore, we obtain the following corollary when we apply the universal lower bound of $\Omega(N)$ for deterministic algorithms from [10] and that of $\Omega(\log N)$ for randomized algorithms from [11] of MTS on every $N$-point metric.

▶ **Corollary 3.22.** *No deterministic algorithm can achieve competitive ratio better than $\Omega(2^n)$ and no randomized algorithm can achieve competitive ratio better than $\Omega(n)$ for MPMD-Size on any $n$-point metric.*

This gives the lower bounds in Theorems 1.1 and 1.2.

# 4 MPMD-Convex on Uniform Metrics

In this section, we study another form of delay functions that does not fall in the category of size-based delays. We use the continuous time model to study the online matching with uniform delays, where the same monotone continuous delay function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ (we further assume that $f(0) = 0$ without loss of generality) applies to every request $r$ in the following way: suppose the arrival time of a request $r$ is $a(r)$ and $t$ is the time when it is matched by the algorithm, then the total delay cost incurred by request $r$ is $f(t - a(r))$. In this section, we still use $\ell(r)$ to denote the location of request $r$ in the metric space, and the connection cost of matching two requests $r_1$ and $r_2$ at locations $\ell(r_1)$ and $\ell(r_2)$ respectively is still $d(\ell(r_1), \ell(r_2))$. The total cost of a solution consists of its total connection cost of matching request into pairs, and the sum of delay cost incurred by every request.

We study monotone convex polynomial delay functions $f$ such that $f'(0) > 0$ on $n$-point uniform metrics where the distance between any two different points is $2\delta > 0$. Equivalently, such a metric can be interpreted as a star metric where leaf nodes are points in the metric space, the distance between every leaf and the root is $\delta$, and the distance between two leaves is the length of the shortest path between them, which is $2\delta$. The previous study by Liu et al. [20] gives a tight $\Theta(n)$ deterministic bound for a class of convex delay functions (including all monotone polynomials) such that $f(0) = f'(0) = 0$, and our study complements the landscape by the following result for functions that $f'(0) > 0$:

▶ **Theorem 4.1.** *For any fixed convex monotone polynomial delay $f$ such that $f'(0) > 0$, there is a constant-competitive deterministic algorithm for MPMD under delay function $f$ on all $n$-point uniform metrics with inter-point distance $\delta$ for constant $\delta$.*

The contrast of this result compared with the $\Omega(n)$ deterministic lower bound of the case of continuous non-decreasing unbounded delays satisfying that $f(0) = f'(0) = 0$ demonstrates that even the local property of the delay function at time $t = 0$ significantly affects the solution and competitiveness of the online MPMD problem. Different from the linear delay case, here the distance $2\delta$ between different points in the metric space also affects the competitive ratio: our algorithm is constant competitive under a fixed function $f$ when $\delta$ is considered as a constant (the same as the study by Liu et al. [20]), while the competitive ratio can also be regarded as a function of $\delta$, which provides another parameter that might be interesting to study for convex delays.

## 4.1 Simpler Polynomials

Without loss of generality, we study $f$ in the form $f(t) = \frac{1}{k}t^k + t$ for $k \geq 2$. To see why we can make this assumption, suppose that we have another polynomial $h(t) = \sum_{i=1}^{k} a_i t^i$ that is monotone on $\mathbb{R}_{\geq 0}$, such that $a_k > 0, a_1 > 0$. [3] The following claim proves that $h$ can be approximated by $f$ within constant factors.

▷ Claim 4.2. There exist constants $A, B > 0$ such that $A \leq \frac{f(t)}{h(t)} \leq B$ for every $t \geq 0$.

Proof. Consider the rational function $\frac{f(t)}{h(t)}$. Since $h(0) = 0$ and $h$ is monotone on $\mathbb{R}_{\geq 0}$, it holds that $h(t) > 0$ for every $t > 0$. As a result, $\frac{f(t)}{h(t)}$ is well-defined for every $t \in (0, +\infty)$.

Moreover, it holds that $\lim_{t\to 0^+} \frac{f(t)}{h(t)} = \frac{1}{a_1}$ according to L'Hôpital's rule and $\lim_{t\to+\infty} \frac{f(t)}{h(t)} = \frac{1}{ka_k}$, and both values are strictly bigger than zero based on our assumption that $a_k$ and $a_1$ are both non-zero values and $h(t)$ is monotone on $\mathbb{R}_{\geq 0}$. Based on the definition of the

[3] We need $a_1 > 0$ to have $h'(0) > 0$.

limit, there exists $T > 0$ such that $0 < \frac{1}{2ka_k} < \frac{f(t)}{h(t)} < \frac{2}{ka_k}$ for every $t > T$. Because $\frac{f(t)}{h(t)}$ is a continuous positive function on the closed interval $[0, T]$ (define $\frac{f(0)}{h(0)} = \frac{1}{a_1}$), it has positive upper and lower bounds on $[0, T]$. Combined with the positive upper and lower bounds on $(T, +\infty)$, we conclude that $\frac{f(t)}{h(t)}$ has positive upper and lower bounds on $\mathbb{R}_{\geq 0}$. ◁

As a result, if there is a competitive algorithm for delay function $f$, then the same algorithm can be applied to delay function $h$ and the lost factor of the competitive ratio is within a constant factor that only depends on $A$ and $B$, independent of both $n$ and $\delta$.

If we apply this idea of comparing the costs of the same solution under different delay functions to comparing the convex delay function $f(t) = \frac{1}{k}t^k + t$ with the classical linear delay function $g(t) = t$ on the same request sequence, the only difference that can lead to different costs and different optimal solutions are delay functions $f$ and $g$. In the following part, we focus on the linear delay function $g$, and prove that a competitive algorithm for $g$ with certain properties also works well for $f$. Given an algorithm $\mathcal{A}$ and some delay function $h$, we use $\text{cost}_{\mathcal{A}}(h)$ to denote the cost of $\mathcal{A}$ when the delay function is set to $h$. As the total cost consists of connection cost and delay cost, we use $\text{cost}^d_{\mathcal{A}}(h)$ to denote its connection cost and $\text{cost}^t_{\mathcal{A}}(h)$ to denote its delay cost. Similarly for any solution SOL, we use $\text{SOL}(h)$ to denote its total cost under $h$, as well as $\text{SOL}^d(h)$ and $\text{SOL}^t(h)$ to denote its connection and delay costs respectively. These are the notations that we will use in computations in the next section.

## 4.2 $T$-Impatient Algorithms

For an online algorithm $\mathcal{A}$ and a request $r$, define $t_{\mathcal{A}}(r)$ to be the time that $r$ is matched by $\mathcal{A}$.

▶ **Definition 4.3** ($T$-impatient algorithms). *An online algorithm $\mathcal{A}$ is* $T$-impatient *for a fixed value $T > 0$ if for every request $r$ such that $t_{\mathcal{A}}(r) - a(r) > T$, $r$ is the only pending request in the whole metric space at every moment throughout the whole time interval $[a(r) + T, t_{\mathcal{A}}(r))$, i.e. all previous requests have been matched and no new requests arrive during the interval.*

Intuitively, a $T$-impatient algorithm will only leave a request $r$ unmatched for more than $T$ time only if there is no other request it can match $r$ to. Equivalently, if a request $r$ has been pending for time $T$, then $\mathcal{A}$ is going to match $r$ with another request immediately as long as there is such a request available. The $T$-impatient property guarantees a lower bound of the optimal solution during the period when there is a request pending for longer than $T$. On the other hand, when no request is pending for longer than $T$, the delay increment can be approximated by the linear delay up to a constant factor as long as $f'(0) > 0$. These two properties guarantee the competitiveness of a $T$-impatient algorithm under convex delays.

The next lemma is the key to guaranteeing that the cost of a $T$-impatient algorithm during the period when there is a request that has been pending beyond time $T$ is within a constant factor of the cost incurred by the optimal solution during the same period.

▶ **Lemma 4.4.** *Given $f(t) = \frac{1}{k}t^k + t$ and a constant $T > 0$, $\frac{f(x)-f(T)}{f(x-T)} < 2^k(T^{k-1} + 1)$ for every $x > T$.*

We now formally state the reduction from convex delay $f(t) = \frac{1}{k}t^k + t$ to linear delay $g(t) = t$ under any $T$-impatient algorithm.

▶ **Lemma 4.5.** *On every $n$-point uniform metric, if a $T$-impatient algorithm $\mathcal{A}$ is $\gamma$-competitive ($\gamma \geq 1$) under delay function $g(t) = t$, then $\mathcal{A}$ is $2^{k+1}(T^{k-1} + 1)\gamma$-competitive under delay function $f(t) = \frac{1}{k}t^k + t$.*

The proofs of Lemmas 4.4 and 4.5 can be found in the full version.

With this reduction, now the goal is to find an $T$-impatient algorithm for some constant $T$ independent of $n$ that achieves competitive ratio $\gamma$ under delay function $g(t) = t$, where $\gamma$ is a constant independent of $n$ as well.

▶ Remark 4.6. At this stage, it is natural to ask whether the previous algorithm that works well for the linear delay on a tree from [2] (as uniform metrics are a special class of trees with height 1 when we view the graph as a star) satisfies the $T$-impatient property. Unfortunately, this is not true and we do have to propose a new algorithm to be applied to convex delays. See the full version for a brief discussion on how their algorithm runs on a uniform star metric, and why it does not satisfy the $T$-impatient property for any constant $T$ even when $\delta$ is a constant.

## 4.3 Constructing a $T$-Impatient Algorithm

In this section, we propose a $T$-impatient algorithm and prove that it is competitive under linear delay $g(t) = t$, and thus a competitive algorithm under $f(t) = \frac{1}{k}t^k + t$ according to Lemma 4.5.

**High-level description.** Recall that a uniform metric can be regarded as a star where each leaf corresponds to a point in the metric space, and the distance between each leaf and the root is $\delta$. We assign a counter to every vertex of the star graph, i.e. all leaves and the root.

During the execution of our algorithm (Algorithm 2), each request can be in one of three states: LEAF, ROOT and READY. A request $r$ is in LEAF if and only if it is contributing to the increase of the leaf counter $z_{\ell(r)}$ ($\ell(r)$ is the location of $r$), in ROOT if it is neither in LEAF nor in READY. If we want to view the dynamic of a request moving between states, then after a request $r$ arrives, initially it is in LEAF until $z_{\ell(r)}$ is saturated (if $z_{\ell(r)}$ is already saturated at arrival time $a(r)$, we can still say that the time length when $r$ is in LEAF is zero). Once $z_{\ell(r)}$ is saturated, $r$ transits to ROOT, and will only transit from ROOT to READY if the condition of either Case 1 or Case 2 is satisfied. The state transition diagram LEAF $\to$ ROOT $\to$ READY is acyclic, which means that once a request $r$ transits to the next state, there is no way back to the previous state. Based on the fact that the counter $z_{\ell(r)}$ is saturated is equivalent to that $r$ is in ROOT or READY for a pending request $r$, the conditions to trigger a match between two requests at different points can also be expressed as: either both are in ROOT, or one is in READY.

Note that the priority rule guarantees that if one request is in READY, then it should be matched immediately as long as there is another pending request, which is vital in guaranteeing the $T$-impatient property.

▷ Claim 4.7. ALG is $4\delta$-impatient.

Proof. It suffices to show that a request $r$ with arrival time $a(r)$ transits to state READY no later than $a(r) + 4\delta$. It takes time at most $\delta$ to transit from LEAF to ROOT and at most $3\delta$ (the sum of Cases 1 and 2) to transit from ROOT to READY after the arrival of a request if it stays unmatched. ◁

**Overview of analysis.** Our analysis is similar to that in [2, 1] in the sense that we analyze counters associated with different points separately, where the time axis can be divided into phases for each counter according to when it is saturated, and we compare the algorithm with another given solution based on *alignment status* (to be defined later). The biggest difference is that their algorithms only match two requests on a uniform metric when both

**Algorithm 2** A Deterministic Algorithm for MPMD on $n$-point uniform metrics with linear delay $g(t) = t$.

**Initialize:** For each leaf vertex $v$, $z_v \leftarrow 0$. For the root vertex $u$, $z_u \leftarrow 0$.

**At every step $t$:**

While there is a request $r$ pending at $\ell(r)$:

1. if $z_{\ell(r)} < \delta$, then $r$ contributes to the increase of the counter $z_{\ell(r)}$ with unit rate and $r$ is in the state LEAF, otherwise if $z_{\ell(r)} = \delta$ then $r$ is in ROOT until it transits to READY;
2. if $z_{\ell(r)} = \delta$ and $r$ is the only pending request, then then $r$ contributes to the increase of the root counter $z_u$;
3. if the single request $r$ has contributed the total value $\delta$ to counter $z_u$, then $r$ transits from ROOT to READY (Case 1);
4. if the total amount of time when $r$ is in ROOT while not contributing to the increase of $z_u$ accumulates to $2\delta$, then $r$ transits from ROOT to READY (Case 2).

**The conditions that trigger a match and counter resetting:**

If there are two pending requests $r_1$ and $r_2$ such that $\ell(r_1) = \ell(r_2)$, match $r_1$ and $r_2$.

If there are two pending requests $r_1$ and $r_2$ such that $z_{\ell(r_1)} = z_{\ell(r_2)} = \delta$, match $r_1$ and $r_2$ and reset both $z_{\ell(r_1)}$ and $z_{\ell(r_2)}$ to 0.

If there are two pending requests $r_1$ and $r_2$ such that $r_1$ is in READY, match $r_1$ and $r_2$ and only reset $z_{\ell(r_1)}$ to 0.

**Priority rule:** If multiple matches can occur at the same time, then the request that transited to state ROOT at the earliest time (as long as it exists) is prioritized.

counters associated with these two requests are saturated, while our algorithm has another case that a request in READY is matched with another request immediately when it appears without waiting for another request's counter to also be saturated. This is the key property that makes Algorithm 2 satisfy the desired $T$-impatient property. To deal with this case for the cost analysis, we prove that the total cost incurred by matches of this case provides a lower bound of the optimal cost as well, hence proving that our new algorithm is also competitive.

Throughout the analysis, we use ALG to denote both the algorithm that we propose and its cost, and SOL to denote a fixed offline solution and its cost that we are comparing ALG with, and the costs are always incurred under delay function $g$. The superscripts $d$ and $t$ that represent connection and delay costs respectively work in the same way as before.

We first note that our algorithm is smart (defined in Section 3.1) as well, which will be used for the alignment status analysis that we are going to introduce.

▶ **Observation 4.8.** *There is at most one request pending at the same point in the metric space at every moment except when a match occurs.*

We can also assume that Observation 4.8 holds for every offline solution SOL, because linear delay is a special case of the size-based delay and Lemma 3.1 holds for this case as well. Our delay function is continuous, which means that the instantaneous delay increment at the discrete moments when a match occurs is negligible. As a result, if we want to compare the status of pending requests under ALG with another solution SOL, at every moment there are four cases at every point $v \in V$: there is one request pending at $v$ under both ALG and SOL, there is no request pending at $v$ under both ALG and SOL, there is one request pending at $v$ under ALG but no request pending at $v$ under SOL, there is one request pending at $v$ under SOL but no request pending at $v$ under ALG. We say that a point $v \in V$ is *aligned* if it is one of the first two cases among the four, otherwise $v$ is *misaligned*.

▶ **Observation 4.9.** *The alignment status of a point $v \in V$ changes if and only if exactly one of* ALG *and* SOL *matches $v$ with another point.*

We say that a request $r$ *carries* $v$ if ALG makes a match between $r$ that is in READY and a request at $v \neq \ell(r)$. For such a request $r$, let $t_1(r)$ be the moment when it starts to be in ROOT and $t_2(r)$ be the moment when it starts to be in READY.

▷ Claim 4.10. Suppose that two different requests $r$ and $r'$ both have reached READY status under Algorithm 2, then $[t_1(r), t_2(r)) \cap [t_1(r'), t_2(r')) = \varnothing$.

Proof. Without loss of generality, assume that $t_1(r) \leq t_1(r')$. At the time $t_1(r')$ when $r'$ reaches ROOT, if $t_2(r) > t_1(r')$, i.e., $r$ is still unmatched in ROOT status, then $r$ and $r'$ should have been matched immediately at $t_1(r')$ if no other request matches either of them, a contradiction to that $r, r'$ both have reached READY at some later time. ◁

▶ **Lemma 4.11.** *Suppose that a request $r$ carries some point, then* SOL $|_{[t_1(r),t_2(r))} \geq \delta$.

**Proof.** We discuss different cases by how $r$ transits from states ROOT to READY. By the design of ALG, now we cover cases 1 and 2 separately.

**Case 1.** Every moment when $r$ is contributing to the increase of the root counter $z_u$, $r$ is the only request that is pending, which means that the total number of arrivals of requests in the input sequence so far is an odd number. Therefore, under any solution SOL, there is at least one request pending and hence the delay is increasing with rate at least 1 at every moment when $z_u$ is increasing. By the condition of Case 1 that $r$ has contributed $\delta$ to the increase of $z_u$, the delay cost of SOL during this period must be at least $\delta$ as well.

**Case 2.** It is easy to see that ALG does not match any two requests at different points during $(t_1(r), t_2(r))$, since every such match must involve another request that has just reached state ROOT, while the priority rule would have matched $r$ instead. If SOL makes a match between two requests at different points, then cost of at least $2\delta$ is incurred.

Otherwise, by Observation 4.9, the alignment of every point stays unchanged throughout $(t_1(r), t_2(r))$. Let $\mathbf{t} = t_2(r) - t_1(r)$ denote the length of this interval, then $\mathbf{t} \geq 2\delta$ by the condition of Case 2 that $r$ transits into READY. For all other points in the metric than $\ell(r)$ denoted by $v_1, \ldots, v_{n-1}$ respectively, let $x_1, \ldots, x_{n-1}$ denote the increased counter value of $v_1, \ldots, v_{n-1}$ during this period respectively, then $x_i \leq \delta$ for every $1 \leq i \leq n-1$, otherwise some counter $z_{v_i}$ should have been saturated since no counter is reset in this period and $r$ should have been matched according to the priority rule. Moreover, $x_1 + \ldots + x_{n-1} \geq 2\delta$, since the total time when there is at least one counter increasing among $v_1, \ldots, v_{n-1}$ has reached $2\delta$ for $r$ to transit into READY. By the definition of the alignment status, for every $v_i$, either it is aligned and therefore there is a request pending at it throughout the total time $x_i$ when $z_{v_i}$ is increasing, or it is misaligned and therefore there is a request pending at it throughout the total time $\mathbf{t} - x_i$ when $z_{v_i}$ is not increasing. In either way, the delay cost incurred by a request pending at $v_i$ is at least $\min\{x_i, \mathbf{t} - x_i\}$, which is equal to $x_i$ by the fact that $x_i \leq \delta$ and $\mathbf{t} \geq 2\delta$. As a result, the total delay time incurred at all points in $v_1, \ldots, v_{n-1}$ is at least $x_1 + \ldots + x_{n-1} \geq 2\delta$.

Therefore, the cost incurred by SOL is at least $\delta$ during $[t_1(r), t_2(r))$ for both cases. ◀

The following corollary is a direct result of combining Claim 4.10 and Lemma 4.11.

▶ **Corollary 4.12.** *Let $R = \{r \mid r$ carries some point during the execution of* ALG$\}$*, then*

$$\text{SOL} \geq \sum_{r \in R} \delta |R|.$$

Now we start to analyze the cost of ALG under the linear delay $g(t) = t$ based on the increments of all leaf counters $z_v$'s and $z_u$. For each leaf counter $z_v$, let $y_v$ denote the total increments of the value of $z_v$ throughout the execution of ALG without any reset. Similarly, let $y_u$ denote the final value of the root counter $z_u$ since the value of $z_u$ is never reset to zero. We refer the reader to the full version of our paper for the proofs of the remaining lemmas in this section.

▶ **Lemma 4.13.** $\mathrm{ALG}^t \le 2\sum_{leaf\ v} y_v + y_u$.

▶ **Lemma 4.14.** $\mathrm{ALG}^d \le 2\sum_{leaf\ v} y_v$.

Now we need to relate the final value $y_u$ of the root counter and $y_v$'s of leaf counters to the cost of any SOL. Given a solution SOL, for every leaf $v$, let $x_v$ denote the delay cost incurred by requests at $v$, $x'_v$ denote the connection cost incurred by matching $v$ with another point (i.e. $x'_v = k\delta$ if $v$ is matched with another point for $k$ times by SOL, and the total connection cost is just the sum among all leaf points). We introduce an additional term based on ALG rather than SOL, which is $b_v$, to denote the number of times that $v$ is *carried* by another request under the execution of ALG.

▶ **Lemma 4.15.** $y_u \le \mathrm{SOL}^t$.

▶ **Lemma 4.16.** $y_v \le 2x_v + x'_v + \delta \cdot b_v$ *for every leaf* $v$.

▶ **Theorem 4.17.** *Algorithm 2 achieves competitive ratio* 13 *under delay function* $g(t) = t$ *and* $13 \cdot 2^{k+1}((4\delta)^{k-1} + 1)$ *under delay function* $f(t) = \frac{1}{k}t^k + t$ *for every* $k \ge 2$.

**Proof.** From Lemmas 4.13 and 4.14, the algorithmic cost $\mathrm{ALG} = \mathrm{ALG}^t + \mathrm{ALG}^d$ is at most $4\sum_{\mathrm{leaf}\ v} y_v + y_u$. By Lemmas 4.15 and 4.16, this is at most $\mathrm{SOL}^t + \sum_{\mathrm{leaf}\ v}(8x_v + 4x'_v + 4\delta \cdot b_v) = 9 \cdot \mathrm{SOL}^t + 4 \cdot \mathrm{SOL}^d + 4\delta \cdot \sum_{\mathrm{leaf}\ v} b_v$, where $b_v$ is the number of times that $v$ is carried by another request. By Corollary 4.12, $\delta \cdot \sum_{\mathrm{leaf}\ v} b_v \le \mathrm{SOL}$. All combined together, we have $\mathrm{ALG} \le 13 \cdot \mathrm{SOL}$, and this is the competitive ratio that ALG achieves under the linear delay function. According to Lemma 4.5, it achieves competitive ratio $13 \cdot 2^{k+1}((4\delta)^{k-1} + t)$ under the delay function $f(t) = \frac{1}{k}t^k + 1$ for $k > 1$. ◀

▶ Remark 4.18. In this section, we use functions of the form $f(t) = \frac{1}{k}t^k + t$ for every $k > 1$ as representatives as all monotone convex polynomials, and show that the competitive ratio can be preserved up to a constant factor independent of $n$ and $\delta$ because of the approximation properties as Claim 4.2. In fact, the $O(1)$-competitive ratio can be achieved by the same algorithm for a broader class of non-negative, non-decreasing convex functions where $f'(0) > 0$, as long as they hold similar properties to Lemma 4.4 that $\frac{f(x)-f(T)}{f(x-T)}$ is upper bounded by a positive value depending on $T$ when $x \to \infty$ for any $T > 0$. This covers every non-negative, non-decreasing and smooth convex function $f$ that $f'(0) > 0$ whose leading term is $\Theta(x^\alpha)$ for any $\alpha > 1$, as well as $\Theta(e^x)$. However, exceptions do exist. For example, $f(x) = e^{x^2} - 1$ does not satisfy this property.

## 5 Discussions and Open Problems

In this paper, we improved both upper and lower bounds of MPMD-Size on every $n$-point metric space. We show that the problem is equivalent to MTS on another $2^{n-1}$-point metric space, and the results are consistent with the state-of-the-art results of MTS. The deterministic competitive ratio is settled up to a constant as the $\Theta(N)$ bound of MTS holds for every $N$-point metric. However, there is still a gap between the upper and lower

bounds for randomized algorithms. This is because the randomized competitive ratio of MTS depends on metric structures: as the randomized competitive ratio of $O(\log N)$ can be achieved on some $N$-point metrics like uniform metrics [10] and weighted stars [12], there also exist $N$-point metrics where the lower bound of $\Omega(\log^2 N)$ holds [11]. As a result, to make further improvements, we need to study the structure of the metric space $(\mathcal{K}^V, D)$. Given a metric space $(V, d)$, the definition of $(\mathcal{K}^V, D)$ is naturally from the Boolean status of every point in $v$, with some similarities to the earthmover distance, yet this structure has not been well studied, and may be of independent interest.

Our study of MPMD-Convex on uniform metrics covers the case $f'(0) > 0$ that has not been considered before, and very surprisingly, such local property at $t = 0$ totally changes the competitiveness compared with the case $f'(0) = 0$. As the majority of smooth convex delay functions have been covered on uniform metrics, the natural open problem is to design an algorithm for convex delays on more general metrics. There are still some open problems on uniform metrics as well: when we consider randomization, there is only a lower bound of $\Omega(\log n)$ for unbounded, non-decreasing and continuous functions that $f(0) = f'(0) = 0$, while it is open whether there is a randomized algorithm that can achieve competitive ratio of $O(\log n)$ in this case. As for the case $f'(0) > 0$, we note that although Algorithm 2 achieves constant competitive ratio when the distance between points is a fixed constant $2\delta$, the competitive ratio depends on $\delta$ and tends to infinity when $\delta$ is sufficiently large. This is different from the linear delay case where the scale of the metric does not affect the competitive ratio. As a result, it might be interesting to study whether it is possible to design an algorithm whose competitive ratio is also independent of $\delta$.

Another natural research direction is to study the bipartite case of MPMD-Size and MPMD-Convex, where every request has either positive or negative polarity and only requests of opposite polarities can be matched. The online min-cost bipartite perfect matching with delay problem has been studied for linear [1] and concave delays [5], yet there is no result for convex or size-based delay. While the proofs and results by Deryckere and Umboh [14] can be easily adapted for the bipartite matching problem by treating every even-sized subset whose sum of polarities is zero as an MTS state, our method for MPMD-Size does not easily extend to the bipartite case as it is not sufficient to express the states with Boolean vectors because multiple requests of the same polarity can accumulate at the same location.

## References


1 Itai Ashlagi, Yossi Azar, Moses Charikar, Ashish Chiplunkar, Ofir Geri, Haim Kaplan, Rahul Makhijani, Yuyi Wang, and Roger Wattenhofer. Min-cost bipartite perfect matching with delays. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 1:1–1:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.1.

2 Yossi Azar, Ashish Chiplunkar, and Haim Kaplan. Polylogarithmic bounds on the competitiveness of min-cost perfect matching with delays. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1051–1061. SIAM, 2017. doi:10.1137/1.9781611974782.67.

3 Yossi Azar, Ashish Chiplunkar, Shay Kutten, and Noam Touitou. Set cover with delay - clairvoyance is not required. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, Pisa, Italy (Virtual Conference), September 7-9, 2020*, volume 173 of *LIPIcs*, pages 8:1–8:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.8.

4 Yossi Azar and Amit Jacob Fanani. Deterministic min-cost matching with delays. *Theory Comput. Syst.*, 64(4):572–592, 2020. doi:10.1007/S00224-019-09963-7.

5 Yossi Azar, Runtian Ren, and Danny Vainstein. The min-cost matching with concave delays problem. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 301–320. SIAM, 2021. doi:10.1137/1.9781611976465.20.

6 Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the $k$-server problem. *J. ACM*, 62(5):40:1–40:49, 2015. doi:10.1145/2783434.

7 Sujoy Bhore, Michał Pawłowski, and Seeun William Umboh. Online tcp acknowledgment under general delays. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2026*, 2026. To appear. Full version available as arXiv:2604.13428 [cs.DS]. doi:10.48550/arXiv.2604.13428.

8 Marcin Bienkowski, Artur Kraska, Hsiang-Hsuan Liu, and Pawel Schmidt. A primal-dual online deterministic algorithm for matching with delays. In Leah Epstein and Thomas Erlebach, editors, *Approximation and Online Algorithms - 16th International Workshop, WAOA 2018, Helsinki, Finland, August 23-24, 2018, Revised Selected Papers*, volume 11312 of *Lecture Notes in Computer Science*, pages 51–68. Springer, 2018. doi:10.1007/978-3-030-04693-4_4.

9 Marcin Bienkowski, Artur Kraska, and Pawel Schmidt. A match in time saves nine: Deterministic online matching with delays. In Roberto Solis-Oba and Rudolf Fleischer, editors, *Approximation and Online Algorithms - 15th International Workshop, WAOA 2017, Vienna, Austria, September 7-8, 2017, Revised Selected Papers*, volume 10787 of *Lecture Notes in Computer Science*, pages 132–146. Springer, 2017. doi:10.1007/978-3-319-89441-6_11.

10 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal online algorithm for metrical task systems. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 373–382. ACM, 1987. doi:10.1145/28395.28435.

11 Sébastien Bubeck, Christian Coester, and Yuval Rabani. The randomized k-server conjecture is false! In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 581–594. ACM, 2023. doi:10.1145/3564246.3585132.

12 Sébastien Bubeck, Michael B. Cohen, James R. Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM J. Comput.*, 50(3):909–923, 2021. doi:10.1137/19M1237879.

13 Ryder Chen, Jahanvi Khatkar, and Seeun William Umboh. Online weighted cardinality joint replenishment problem with delay. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, Paris, France, July 4-8, 2022*, volume 229 of *LIPIcs*, pages 40:1–40:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.40.

14 Lindsey Deryckere and Seeun William Umboh. Online matching with set and concave delays. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, Atlanta, Georgia, USA, September 11-13, 2023*, volume 275 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.17.

15 Marc Dufay and Roger Wattenhofer. A deterministic polylogarithmic competitive algorithm for matching with delays. In Kasper Green Larsen and Barna Saha, editors, *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*, pages 3936–3964. SIAM, 2026. doi:10.1137/1.9781611978971.144.

16 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: haste makes waste! In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 333–344. ACM, 2016. doi:10.1145/2897518.2897557.

**17** Leah Epstein. On bin packing with clustering and bin packing with delays. *Discret. Optim.*, 41:100647, 2021. doi:10.1016/J.DISOPT.2021.100647.

**18** Tomer Ezra, Stefano Leonardi, Michal Pawlowski, Matteo Russo, and Seeun William Umboh. Universal optimization for non-clairvoyant subadditive joint replenishment. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, London School of Economics, London, UK, August 28-30, 2024*, LIPIcs, pages 12:1–12:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.APPROX/RANDOM.2024.12.

**19** Ngoc Mai Le, Seeun William Umboh, and Ningyuan Xie. The power of clairvoyance for multi-level aggregation and set cover with delay. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1594–1610. SIAM, 2023. doi:10.1137/1.9781611977554.CH59.

**20** Xingwu Liu, Zhida Pan, Yuyi Wang, and Roger Wattenhofer. Impatient online matching. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *29th International Symposium on Algorithms and Computation, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, volume 123 of *LIPIcs*, pages 62:1–62:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ISAAC.2018.62.