A Combinatorial Metrical Task System Problem under the Uniform Metric

Nakazono, Takumi
Department of Informatics, Kyushu University

Moridomi, Ken-ichiro
Department of Informatics, Kyushu University

Hatano, Kohei
Library, Kyushu Univerisity

Eiji, Takimoto
Department of Informatics, Kyushu University

A Combinatorial Metrical Task System Problem under the Uniform Metric

Takumi Nakazono[1] [⋆], Ken-ichiro Moridomi[1], Kohei Hatano[2], and Eiji Takimoto[1]
{moridomi.kenichiro, hatano, eiji}@inf.kyushu-u.ac.jp

[1] Department of Informatics, Kyushu University
[2] Library, Kyushu University

Abstract. We consider a variant of the metrical task system (MTS) problem under the uniform metric, where each decision corresponds to some combinatorial object in a fixed set (e.g., the set of all s-t paths of a fixed graph). Typical algorithms such as Marking algorithm are not known to solve this problem efficiently and straightforward implementations takes exponential time for many classes of combinatorial sets. We propose a modification of Marking algorithm, which we call Weighted Marking algorithm. We show that Weighted Marking algorithm still keeps $O(\log n)$ competitive ratio for the standard MTS problem with n states. On the other hand, combining with known sampling techniques for combinatorial sets, Weighted Marking algorithm works efficiently for various classes of combinatorial sets.

1 Introduction

The metrical task system is defined as a repeated game between the player and the adversary. Given a fixed set C of states a metric $\delta : C \times C \to \mathbb{R}_+$ and a initial state $c_0 \in C$, for each round $t = 1, \dots, T$, (i) the adversary reveals a (processing) cost function $f_t : C \to \mathbb{R}_+$, (ii) the player chooses a state $c_t \in C$, and (iii) the player incurs the processing cost $f_t(c_t)$ and the moving cost $\delta(c_t, c_{t-1})$. The goal of the algorithm is minimizing the cumulative (processing and moving) cost. The performance of the algorithm is measured by the competitive ratio, that is, the ratio of the cumulative cost of the algorithm to the cumulative cost of the best fixed sequence of states in hindsight.

In the expert setting, i.e., where the decision set consists of n states, there are many existing works on the MTS [8, 14, 4, 11, 5]. In particular, for the uniform metric δ (which is defined as $\delta(i, j) = 1$ if $i \neq j$ and otherwise $\delta(i, j) = 0$), the MTS problem is well studied [8, 4, 14, 1]. Borodin et al. show the lower bound of the competitive ratio of any randomized algorithm is H_n, where H_n is the n-th harmonic number [8]. Especially, Abernethy et al. provide an algorithm which uses the method of convex optimization, and shows the upper bound of the competitive ratio of the algorithm is $H_n + O(\log \log n)$ [1].

[⋆] Currently with Toshiba Solutions Corporation.

When we consider the situation where the decision set C is a combinatorial set from $\{0,1\}^d$ (e.g., the set of spanning trees or s-t paths of a graph), the computational issue arises. A natural example of a combinatorial MTS is a routing problem. For example, we consider a routing problem. Consider a fixed network $G = (V, E)$ where V is the set of routers (nodes) and $E \subseteq V \times V$ is the set of d edges between routers and V includes two routers, the source s and the sink t. The decision set C is the set of paths from s to t, whose size is exponential in d. In general, for typical combinatorial sets, the size could be exponential in the dimension size d as well and straightforward implementations of known algorithms for the MTS take exponential time as well since time complexity of these algorithms is proportional to the size n of the decision set.

In this paper, for the uniform metric, we propose a modification of the Marking algorithm [8], which we call the Weighted Marking algorithm. The weighted Marking algorithm employs an exponential weighting scheme and can be viewed as an analogue of the Hedge algorithm [12] for the MTS problem, whereas the Marking algorithm is an analogue of the classical Halving algorithm. We prove that the Weighted Marking algorithm retains $O(\log n)$ competitive ratio for the standard MTS problem with n states. The expected running time of the Weighted Marking algorithm at each round is the same as that of the original one.

On the other hand, combining with efficient sampling techniques w.r.t. exponential weights on combinatorial objects (k-sets, s-t paths [18], stars in a graph [10] permutation matrices [10, 15], permutation vectors [2]), the Weighted Marking algorithm works efficiently for various classes of combinatorial sets.

1.1 Related Work

There are some existing works for combinatorial metrical task systems. Blum et al. provide algorithms for the list update problem [6]. For the k-server problem, which can be viewed as a combinatorial MTS problem, Koutsoupias et al. provide a deterministic algorithm [17]. Bansal et al. improve the results of Koutsoupias et al. by a randomization technique [3]. These algorithms are efficient and perform well for specific problems, i.e., the list update problem and the k-server problem. However, these algorithms are specialized for limited decision sets and we cannot use them for other problems.

Buchbinder et al. consider combinatorial MTS problems where the decision space is defined as a matroid [9]. The concept of matroid can express various classes of combinatorial objects such as spanning trees. They show a unified algorithm with a guaranteed competitive ratio. Their analysis is, however, applicable for a continuous "relaxed" space only. It is not known if there exists a rounding scheme that approximately preserves the moving cost over the relaxed space. Gupta et al. also consider combinatorial MTS problems over the basis of a matroid [13]. They give a rounding algorithm and prove the competitive ratio of a rounded solution, for a class of metrics including the Hamming distance but not the uniform metric.

2 Preliminaries

A metrical task system (MTS) is a pair (C, δ) where C is a set of states and $\delta : C \times C \to \mathbb{R}_+$ is a metric. In particular, we consider a combinatorial setting where C is a subset of $\{0,1\}^d$ for some dimension $d > 0$. We denote by n the size of C, that is, $n = |C|$. Typically, n is exponentially large in d. Moreover, we only consider the uniform metric δ, that is,

$$\delta(\boldsymbol{c}_1, \boldsymbol{c}_2) = \begin{cases} 1 & \text{if } \boldsymbol{c}_1 \neq \boldsymbol{c}_2, \\ 0 & \text{if } \boldsymbol{c}_1 = \boldsymbol{c}_2. \end{cases}$$

The combinatorial MTS problem for (C, δ) is defined as the following protocol between the algorithm and the adversary.

First the adversary chooses a task sequence $\sigma = (\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_T)$, where each $\boldsymbol{\ell}_t \in [0,1]^d$ is called a loss vector. In other words, we assume the oblivious setting. For a given initial state $\boldsymbol{c}_0 \in C$, the protocol proceeds in rounds, where in each round $t = 1, 2, \ldots, T$,

1. the algorithm receives the loss vector $\boldsymbol{\ell}_t \in [0,1]^d$,
2. the algorithm chooses a state $\boldsymbol{c}_t \in C$, and
3. the algorithm suffers a cost given by $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t + \delta(\boldsymbol{c}_t, \boldsymbol{c}_{t-1})$.

The first term $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$ of the cost is called the processing cost at round t, and the second term $\delta(\boldsymbol{c}_t, \boldsymbol{c}_{t-1})$ is called the moving cost at round t.

For a task sequence σ, the cumulative cost of an algorithm A is defined as

$$\mathrm{cost}_A(\sigma) = \sum_{t=1}^{T} (\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t + \delta(\boldsymbol{c}_t, \boldsymbol{c}_{t-1})),$$

and the cumulative cost of the best offline solution is defined as

$$\mathrm{cost}_{\mathrm{OPT}}(\sigma) = \min_{(\boldsymbol{c}_1^*, \boldsymbol{c}_2^*, \ldots, \boldsymbol{c}_T^*) \in C^T} \sum_{t=1}^{T} (\boldsymbol{c}_t^* \cdot \boldsymbol{\ell}_t + \delta(\boldsymbol{c}_t^*, \boldsymbol{c}_{t-1}^*)).$$

We measure the performance of algorithm A by its competitive ratio, which is defined as

$$\mathrm{CR}(\sigma) = \frac{\mathbb{E}[\mathrm{cost}_A(\sigma)]}{\mathrm{cost}_{\mathrm{OPT}}(\sigma)},$$

where the expectation is with respect to the internal randomness of A. The goal of the algorithm is to minimize the worst case competitive ratio $\max_\sigma \mathrm{CR}(\sigma)$. Note that the usual (non-combinatorial) MTS problem is a special case where C consists of unit vectors.

We also require the algorithm A to produce a state $\boldsymbol{c}_t$ in time polynomial in d for each round t. Typically, the size n of C is exponential in d, and so we cannot directly maintain all states c in C. Therefore, we assume two oracles to access the state set C efficiently. The first one is the linear optimization oracle, which solves the following decision problem:

OPT(C)

Input: $\boldsymbol{L} \in \mathbb{R}_+^d$

Output: $\begin{cases} 0 & \text{if } \min_{\boldsymbol{c} \in C} \boldsymbol{c} \cdot \boldsymbol{L} < 1, \\ 1 & \text{otherwise.} \end{cases}$

The assumption of this oracle is natural since the linear optimization problem has a polynomial time algorithm for many useful state sets C.

The second one is a sampling oracle, which chooses a state $\boldsymbol{c}$ randomly according to a certain probability distribution over C, where the distribution is specified by a given parameter $\boldsymbol{L} \in \mathbb{R}_+^d$. In particular, we consider two kinds of sampling oracles, which will be defined later.

3 The Marking algorithm

Here we apply the Marking algorithm [8] to the combinatorial MTS problem. The Marking algorithm is a simple randomized algorithm whose competitive ratio is upper bounded by $2H_n \leq 2(\ln n + 1)$, where H_n is the n-th harmonic number.

Below we describe how the Marking algorithm works. For a naive implementation, it maintains the cumulative processing costs $l[\boldsymbol{c}]$ for all states $\boldsymbol{c} \in C$. For each round t,

1. Observe the loss vector $\boldsymbol{\ell}_t$ and update $l[\boldsymbol{c}] = l[\boldsymbol{c}] + \boldsymbol{c} \cdot \boldsymbol{\ell}_t$ for all $\boldsymbol{c} \in C$.
2. If $l[\boldsymbol{c}_{t-1}] < 1$ then output $\boldsymbol{c}_t = \boldsymbol{c}_{t-1}$.
3. Else choose a state $\boldsymbol{c}_t$ uniformly at random from the set of states $\boldsymbol{c}$ with $l[\boldsymbol{c}] < 1$, and output $\boldsymbol{c}_t$.
4. If no such states exist, then reset $l[\boldsymbol{c}] = 0$ for all $\boldsymbol{c} \in C$ and choose a state $\boldsymbol{c}_t$ uniformly at random from C, and output $\boldsymbol{c}_t$.

Note that Line 2 and Line 3 intuitively mean that the Marking algorithm does not change states until $l[\boldsymbol{c}_t] \geq 1$. As is well known as a folklore (See, e.g., [7]), we can assume without loss of generality that the loss vectors $\boldsymbol{\ell}_t$ are small enough so that $l[\boldsymbol{c}_t] \leq 1$ always holds. In the appendix we give more detailed discussion. In other words, the Marking algorithm changes states only when $l[\boldsymbol{c}_t] = 1$.

Of course, the naive implementation of the Marking algorithm is not efficient because it maintains the cumulative processing cost $l[\boldsymbol{c}]$ for all states $\boldsymbol{c} \in C$. Instead, we can maintain the cumulative loss vector $\boldsymbol{L} = \sum_t \boldsymbol{\ell}_t$, which implicitly maintains $l[\boldsymbol{c}]$ as $l[\boldsymbol{c}] = \boldsymbol{c} \cdot \boldsymbol{L}$ for all $\boldsymbol{c}$. Furthermore, the sampling problem at Line 3 can be restated as the following problem in terms of $\boldsymbol{L}$, which we call Sampling 1.

Sampling 1

Input: $\boldsymbol{L} \in \mathbb{R}_+^d$,

Output: $\boldsymbol{c} \in C_{\boldsymbol{L}} = \{\boldsymbol{c} \in C \mid \boldsymbol{c} \cdot \boldsymbol{L} < 1\}$ uniformly at random.

Note that the problem Sampling 1 is only defined when $C_{\boldsymbol{L}} \neq \emptyset$, but we can check whether the condition holds by using the linear optimization oracle for $\mathrm{OPT}(C)$. Moreover, the uniform sampling at Line 4 is also restated as Sampling 1 with $\boldsymbol{L} = \boldsymbol{0}$. So, if we assume a linear optimization oracle for $\mathrm{OPT}(C)$ and a sampling oracle for Sampling 1, then we can emulate the Marking algorithm in $O(d)$ time per round. We give this implementation of the Marking algorithm in Algorithm 1.

Algorithm 1 An implementation of the Marking algorithm

```
Input: A linear optimization oracle for $\mathrm{OPT}(C)$ and a sampling oracle for Sampling 1
Initialize: Let $\boldsymbol{L} = \boldsymbol{0}$.
For each round $t = 1, 2, \ldots, T$,
  1. Observe the loss vector $\boldsymbol{\ell}_t$ and update $\boldsymbol{L} = \boldsymbol{L} + \boldsymbol{\ell}_t$.
  2. Let $\boldsymbol{c}_t = \boldsymbol{c}_{t-1}$ and output $\boldsymbol{c}_t$.
  3. If $\boldsymbol{c}_t \cdot \boldsymbol{L} \geq 1$, then
     (a) If $\min_{\boldsymbol{c} \in C} \boldsymbol{c} \cdot \boldsymbol{L} \geq 1$, then reset $\boldsymbol{L} = \boldsymbol{0}$.   // use the linear optimization oracle
     (b) Choose a state $\boldsymbol{c}_t \in C_{\boldsymbol{L}}$ uniformly at random.   // use the sampling oracle
```

The question that naturally arises is that for what state set C, the problem Sampling 1 is efficiently solved. Unfortunately, we do not know any non-trivial sets C that have polynomial time algorithm for Sampling 1. We could use MCMC sampling methods to design approximate sampling, but it seems hard to show theoretically guaranteed performance bounds for many natural state sets C.

4 The Weighted Marking algorithm

The computational cost of the sampling problem Sampling 1 would be due to the fact that the support of the sampling distribution is restricted to the set $C_{\boldsymbol{L}}$. So, we relax the distribution to a continuous distribution whose support is not restricted to $C_{\boldsymbol{L}}$.

Specifically, we propose the following sampling problem, called Sampling 2.

<u>Sampling 2</u>

Input: $\boldsymbol{L} \in \mathbb{R}_+^d$,

Output: $\boldsymbol{c} \in C$ chosen with probability $\pi_{\boldsymbol{L}}(\boldsymbol{c}) = \dfrac{\exp(-\eta \boldsymbol{c} \cdot \boldsymbol{L})}{\sum_{\boldsymbol{c} \in C} \exp(-\eta \boldsymbol{c} \cdot \boldsymbol{L})}$,

where $\eta > 0$ is a parameter.

In words, the new sampling distribution $\pi_{\boldsymbol{L}}$ is such that $\pi_{\boldsymbol{L}}(\boldsymbol{c})$ is a monotone decreasing function with respect to its cumulative processing cost $l[\boldsymbol{c}] = \boldsymbol{c} \cdot \boldsymbol{L}$. So, the probability that a state $\boldsymbol{c}$ with large $l[\boldsymbol{c}]$ is chosen is very low, and thus we will see that the support of $\pi_{\boldsymbol{L}}$ is essentially restricted to a set $\{\boldsymbol{c} \in C \mid \boldsymbol{c} \cdot \boldsymbol{L} < L\}$ for some $L > 1$.

Unlike Sampling 1, there are known efficient implementations of Sampling 2 for several combinatorial objects such as k-sets, s-t paths [18], permutation matrices [10, 15], stars in a graph [10] and permutation vectors [2].

Now we modify the Marking algorithm by assuming the sampling oracle for Sampling 2, as well as assuming the linear optimization oracle for $\mathrm{OPT}(C)$. The modified version is called the Weighted Marking algorithm. The difference from the Marking algorithm is that (1) it does not change states until its cumulative processing cost reaches L instead of 1, and (2) it uses $\pi_{\boldsymbol{L}}$ as the sampling distribution instead of the uniform distribution over $C_{\boldsymbol{L}}$. Note that the Weighted Marking algorithm resets the cumulative loss vector as $\boldsymbol{L} = \boldsymbol{0}$ when $\min_{\boldsymbol{c} \in C} \boldsymbol{c} \cdot \boldsymbol{L}$ reaches 1, which is the same condition as the Marking algorithm. So, unlike the Marking algorithm, resetting $\boldsymbol{L}$ may happen at some round where the cumulative processing cost of the current state does not reach L, since $L \neq 1$.

The detailed description of the Weighted Marking algorithm is given in Algorithm 2.

Algorithm 2 Weighted Marking algorithm

```
Input: A linear optimization oracle for $\mathrm{OPT}(C)$ and a sampling oracle for Sampling 2
Parameter: $\eta > 0$ and $L > 1$ such that $ne^{-\eta L} \leq e^{-\eta}/2$.
Initialize: Let $\boldsymbol{L} = \boldsymbol{0}$.
For each round $t = 1, 2, \ldots, T$,

  1. Observe the loss vector $\boldsymbol{\ell}_t$ and update $\boldsymbol{L} = \boldsymbol{L} + \boldsymbol{\ell}_t$.
  2. Let $\boldsymbol{c}_t = \boldsymbol{c}_{t-1}$ and output $\boldsymbol{c}_t$.
  3. If $\min_{\boldsymbol{c} \in C} \boldsymbol{c} \cdot \boldsymbol{L} \geq 1$ then          // use the linear optimization oracle
     (a) Reset $\boldsymbol{L} = \boldsymbol{0}$.
     (b) Choose a state $\boldsymbol{c}_t \in C$ with probability $\pi_{\boldsymbol{L}}(\boldsymbol{c})$     // use the sampling oracle
  4. Else if $\boldsymbol{c}_t \cdot \boldsymbol{L} \geq L$, then
     (a) Repeat
           Choose a state $\boldsymbol{c}_t \in C$ with probability $\pi_{\boldsymbol{L}}(\boldsymbol{c})$     // use the sampling oracle
         Until $\boldsymbol{c}_t \cdot \boldsymbol{L} < L$.
```

For convenience, we define the notion of phases for analyzing the behavior of the Weighted Marking algorithm. A phase is an interval $\{t \mid t_b \leq t \leq t_e\}$ of rounds such that the resetting happens at round $t_b - 1$ and t_e but does not happen at every round $t_b \leq t < t_e$.

Again, as is well known as a folklore, we assume without loss of generality that the loss vectors $\boldsymbol{\ell}_t$ are small enough so that it always holds that $\min_{\boldsymbol{c} \in C} \boldsymbol{c} \cdot \boldsymbol{L} \leq 1$ at Line 3 and it always hold that $\boldsymbol{c}_t \cdot \boldsymbol{L} \leq L$ at Line 4. In other words, a phase ends (resetting happens) only when $\min_{\boldsymbol{c} \in C} \boldsymbol{c} \cdot \boldsymbol{L} = 1$ and states $\boldsymbol{c}_t$ are changed only when $\boldsymbol{c}_t \cdot \boldsymbol{L} = L$. These assumptions greatly simplifies the analysis.

More formally, the assumption is described as follows:

Assumption 1 *Whenever the previous state* $\mathbf{c}_{t-1}$ *satisfies* $\mathbf{c}_{t-1} \cdot \boldsymbol{L} < L$*, where* $\boldsymbol{L}$ *is the cumulative loss vectors up to round* $t-1$ *in the current phase, and the* *phase did not end at round $t-1$, i.e.,* $\min_{\boldsymbol{c}^*\in C} \boldsymbol{c}^* \cdot \boldsymbol{L} < L$, *then* $\boldsymbol{\ell}_t$ *satisfies the two conditions:*

1. $\boldsymbol{c}_{t-1} \cdot (\boldsymbol{L} + \boldsymbol{\ell}_t) \leq L$, *and*
2. $\min_{\boldsymbol{c}^*\in C} \boldsymbol{c}^* \cdot (\boldsymbol{L} + \boldsymbol{\ell}_t) \leq 1$.

We assume Assumption 1 holds throughout this section. In the appendix, we briefly explain why the assumption holds without loss of generality.

In the next theorem, we give an upper bound of the competitive ratio of the Weighted Marking algorithm.

Theorem 1. *Let* $\eta = \ln 2n$, *and* $L = 2$. *Then for any task sequence* $\sigma = (\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \ldots, \boldsymbol{\ell}_T)$, *the competitive ratio of the Weighted Marking algorithm is upper bounded by*

$$\mathrm{CR}(\sigma) \leq 6e \ln n + 9.$$

Moreover, the expected running time per round is $O(d + T_{\mathrm{lin}} + T_{\mathrm{Samp2}})$, *where* T_{lin} *is the running time of the linear optimization oracle and* T_{Samp2} *is that of the sampling oracle for Sampling 2.*

To prove this theorem, we show that the cumulative moving cost in each phase is $O(\log n)$. So in the following, we fix a particular phase $I = \{t_b, \ldots, t_e\}$. For each round $t \in I$, $\boldsymbol{L}_t$ denotes the cumulative loss vector $\boldsymbol{L}$ at Line 1 at round t. Note by definition that $\min_{\boldsymbol{c}^*\in C} \boldsymbol{c}^* \cdot \boldsymbol{L}_{t_e} = 1$.

Let $G = \{\boldsymbol{c} \in C \mid \boldsymbol{c} \cdot \boldsymbol{L}_{t_e} < L\}$ be the goal set, meaning that if we choose a state in G at some round $t \in I$, i.e., $\boldsymbol{c}_t \in G$, then the Weighted Marking algorithm never changes the state until the end of the phase. Note that $\boldsymbol{c}^* \in G$ and so $G \neq \emptyset$. Let $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n$ be the members of C. (This is an abuse of notation. Do not confuse them with the states $\boldsymbol{c}_t$ the algorithm chooses at round t.) For any $\boldsymbol{c}_i \notin G$, we can define $t_i \in I$ such that $\boldsymbol{c}_i \cdot \boldsymbol{L}_{t_i} = L$. Then, without loss of generality, we assume $t_1 \leq t_2 \leq \cdots \leq t_{n-|G|}$ and $\boldsymbol{c}_n = \boldsymbol{c}^*$, i.e., $\boldsymbol{c}_n \cdot \boldsymbol{L}_{t_e} = 1$. Moreover, we assume $|G| = 1$ just for simplicity. Clearly, the algorithm changes states only at some rounds in $\{t_1, \ldots, t_{n-1}\}$. Let $t^{(k)}$ be the round where the algorithm makes the k-th change of states. For any state $\boldsymbol{c} \in C$, we define the weight function $W_k(\boldsymbol{c})$ as

$$W_k(\boldsymbol{c}) := \begin{cases} e^{-\eta \boldsymbol{c}\cdot \boldsymbol{L}_{t^{(k)}}} & \text{if } \boldsymbol{c} \cdot \boldsymbol{L}_{t^{(k)}} < L, \\ 0 & \text{if } \boldsymbol{c} \cdot \boldsymbol{L}_{t^{(k)}} \geq L. \end{cases}$$

Let $\overline{W}_k := \sum_{\boldsymbol{c}\in C} W_k(\boldsymbol{c})$. Then $W_k(\boldsymbol{c})/\overline{W}_k$ is the probability of choosing state $\boldsymbol{c}$ at the k-th change of states. One can see that $W_k(\boldsymbol{c})$ is monotonically decreasing w.r.t. k because L_t is monotonically increasing vector w.r.t. t.

If the best offline solution changes its state in the phase, then its cumulative moving cost is at least 1, and otherwise its cumulative processing cost is at least 1 by the definition of the phase. This immediately implies the following lemma.

Lemma 2. *For any sequence of loss vectors* $(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \cdots, \boldsymbol{\ell}_T)$, *the best offline solution suffers cost at least* 1 *on each phase.*

On the other hand, whenever the Weighted Marking algorithm changes states (i.e., suffers the moving cost of 1) from $\boldsymbol{c}_{t^{(k-1)}}$ to $\boldsymbol{c}_{t^{(k)}}$, then its cumulative processing cost from $t^{(k-1)}$ to $t^{(k)}$ is at most L. This implies the following lemma.

Lemma 3. *For any sequence of loss vectors $(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \cdots, \boldsymbol{\ell}_T)$, the cumulative processing cost of the Weighted Marking algorithm is at most L times the cumulative moving cost on each phase.*

The following lemma provides the probability of ending a phase.

Lemma 4. *For any $\alpha \in (0,1)$ and for any k, if $\alpha\overline{W}_k \leq e^{-\eta}$ holds then the phase will end at the $k+1$-th change of the state with probability at least α.*

Proof. By the assumption $\boldsymbol{c}_n \cdot \boldsymbol{L}_{t_e} = 1$, if the algorithm choose $\boldsymbol{c}_n$ then the algorithm will change its state at the end of the phase t_e, i.e. if the state $\boldsymbol{c}_n$ is chosen then the phase rests only 1 change. By $\boldsymbol{c}_n \cdot \boldsymbol{L} \leq 1$, we get $W_k(\boldsymbol{c}_n) \geq e^{-\eta}$ for any k. Using this and the condition of the lemma, we get

$$\alpha \leq \frac{e^{-\eta}}{\overline{W}_k} \leq \frac{W_k(\boldsymbol{c}_n)}{\overline{W}_k}.$$

Here, the right hand side is the probability of the state $\boldsymbol{c}_n$ will be chosen by the Weighted Marking algorithm. □

The following lemma guarantees the probability of choosing $\boldsymbol{c}_n$ becomes higher at each change of the state.

Lemma 5. *For any $\alpha \in (0,1)$, for any k, if $\alpha\overline{W}_k \geq e^{-\eta}$ holds then*

$$\Pr[\overline{W}_{k+1} \leq \alpha\overline{W}_k] > \alpha.$$

Proof. Summing up weights of states from $n, n-1, \cdots$ and consider when the sum gets greater than $\alpha\overline{W}_k$. E.g. consider i_k s.t. $\sum_{i=i_k+1}^{n} W_k(\boldsymbol{c}_i) \leq \alpha\overline{W}_k$ and $\sum_{i=i_k}^{n} W_k(\boldsymbol{c}_i) > \alpha\overline{W}_k$.

Assume that the Weighted Marking algorithm chooses the state $\boldsymbol{c}_s$ at the k-th change of the state. If $s \geq i_k$, the algorithm changes its state at $t^{(k+1)}$ and then $W_{k+1}(\boldsymbol{c}_i) = 0$ for any $i \geq i_k$ by the definition of W and i_k. Thus,

$$\overline{W}_{k+1} = \sum_{i=1}^{n} W_{k+1}(\boldsymbol{c}_i) = \sum_{i=i_k+1}^{n} W_{k+1}(\boldsymbol{c}_i).$$

Because W_k is monotonically decreasing w.r.t. k, one can get

$$\sum_{i=i_k+1}^{n} W_{k+1}(\boldsymbol{c}_i) \leq \sum_{i=i_k+1}^{n} W_k(\boldsymbol{c}_i) \leq \alpha\overline{W}.$$

So we get if $s \geq i_k$ then $\overline{W}_{k+1} \leq \alpha\overline{W}$. The probability of the Weighted Marking algorithm choosing the state $\boldsymbol{c}_s$ such that $s \geq i_k$ satisfies

$$\Pr[s \geq i_k] = \frac{\sum_{i=i_k}^{n} W_k(\boldsymbol{c}_i)}{\overline{W}_k} > \frac{\alpha\overline{W}_k}{\overline{W}_k} = \alpha.$$

□

By Lemma 4, one can get the following immediately.

Lemma 6. *For any $\alpha \in (0,1)$ and round $t^{(k)}$, if $\alpha\overline{W}_k \le e^{-\eta}$ then the expected number of remaining changes of states in the phase is less than $\frac{1}{\alpha}+1$.*

Because of W_k is monotonically decreasing w.r.t. k and Lemma 5, one can get the following lemma.

Lemma 7. *For any k, for any $\alpha \in (0,1)$, if $\alpha\overline{W}_k \le e^{-\eta}$ then the expectation of m such that $\overline{W}_{k+m} \le \alpha\overline{W}_k$ is $\mathbb{E}[m] < \frac{1}{\alpha}$.*

We say that a sequence $\overline{W} = \{\overline{W}_1, \overline{W}_2, \cdots, \overline{W}_K\}$ of weights is α-fast decreasing at the round $t^{(k+1)}$ if $\overline{W}_{k+1} \ge \alpha\overline{W}_k$ holds.

Proof (Proof of Theorem 1). Assume that the Weighted Marking algorithm changes its state at K times in a phase. By Lemma 6, if $\alpha\overline{W}_{k'} \le e^{-\eta}$ holds then we have

$$\mathbb{E}[K] \le k' + \frac{1}{\alpha} + 1.$$

Thus, we need to estimate k' s.t. $\alpha\overline{W}_{k'} \le e^{-\eta}$ to bound $\mathbb{E}[K]$.

Let $\alpha\overline{W}_{k'} \le e^{-\eta}$ holds after α-fast decreasing K' times, then

$$\alpha^{K'}\overline{W}_0 \le \overline{W}_{k'} \le \frac{e^{\eta}}{\alpha}.$$

By $\overline{W}_0 = n$, we get $\alpha^{K'} n \le \frac{e^{-\eta}}{\alpha}$ and rearranging, $K' \le \frac{1}{\ln\frac{1}{\alpha}}(\ln n + \eta) - 1$. Using Lemma 7,

$$\mathbb{E}[k'] \le \mathbb{E}[m]K' = \frac{1}{\alpha}K' = \frac{1}{\alpha\ln\frac{1}{\alpha}}(\ln n + \eta) - \frac{1}{\alpha}.$$

Thus, the number of changing of states at a phase is

$$\mathbb{E}[K] \le \mathbb{E}[k'] + \frac{1}{\alpha} + 1 = \frac{1}{\alpha\ln\frac{1}{\alpha}}(\ln n + \eta) + 1.$$

The bound of $\mathbb{E}[K]$ is minimized when $\alpha = 1/e$. So we get $\mathbb{E}[K] \le e(\ln n + \eta) + 1$.

Setting $\eta = \ln 2n$, we get $\mathbb{E}[K] \le 2e\ln n + 3$. By Lemma 3,

$$\mathbb{E}[(\text{cumulative processing cost})] \le L \times \mathbb{E}[(\text{cumulative moving cost})].$$

At each phase, we have

$$\begin{aligned}
&\mathbb{E}[\text{Cumulative loss}]\\
&= \mathbb{E}[\text{Cumulative processing cost}] + \mathbb{E}[\text{Cumulative moving cost}]\\
&\le 3 \times \mathbb{E}[K]\\
&\le 6e\ln n + 9.
\end{aligned}$$

By Lemma 2, at each phase the best offline solution has the cumulative processing cost at least 1. Thus we get the bound of the competitive ratio. □

Next, we prove the running time of the Weighted Marking algorithm. The key point of analysis of the Weighted Marking algorithm is the number of calls to the oracle for Sampling 2 at Line 4-(a) of the pseudo code. The following lemma gives a theoretical bound of retrying.

Lemma 8. *The expected number of calls to the sampling oracle at Line 4-(a) is at most* 2.

Proof. For any state $\boldsymbol{c}$ such that $\boldsymbol{c} \cdot \boldsymbol{L} \geq L$, the probability that the sampling oracle chooses $\boldsymbol{c}$ is

$$\begin{aligned}\frac{\exp(-\eta \boldsymbol{c} \cdot \boldsymbol{L})}{\sum_{\boldsymbol{c}'} \exp(-\eta \boldsymbol{c}' \cdot \boldsymbol{L})} &\leq \frac{\exp(-\eta L)}{\exp(-\eta \boldsymbol{c}_n \cdot \boldsymbol{L})} \\ &\leq \frac{\exp(-\eta L)}{\exp(-\eta)}\end{aligned}$$

since $\boldsymbol{c}_n \cdot \boldsymbol{L} < 1$. By the union bound, the probability that the sampling oracle chooses some $\boldsymbol{c}$ with $\boldsymbol{c} \cdot \boldsymbol{L} \geq L$ is at most

$$\frac{n \exp(-\eta L)}{\exp(-\eta)} = \frac{1}{2}$$

by our choice of η and L. □

5 Conclusion and future work

In this paper, we proposed the Weighted Marking algorithm for combinatorial MTS problems under the uniform metric space, and proved its competitive ratio is at most $6e \ln n + 9 = O(\log n)$. We showed that, by combining with existing sampling techniques for exponential weights over combinatorial objects, the proposed algorithm runs efficiently for several combinatorial classes, e.g., s-t paths and k-sets.

There are several open problems to investigate. First one is to provide a lower bound of the competitive ratio of the combinatorial MTS. In particular, it still remains open to prove $\Omega(\log d)$ or $\Omega(\log n)$ lower bounds for some combinatorial class of the decision set.

Secondly, it is not known if FPL [16] is applicable for the combinatorial MTS problem. If so, the sampling oracle is no longer necessary and we could efficiently solve MTS problems for more classes of combinatorial objects.

Finally, the hardness of the Sampling $1(C)$ is not known, either. Our conjecture is, it is $\#P$ hard for a specific class.

Acknowledgments

We thank anonymous reviewers for useful comments. Hatano is grateful to the supports from JSPS KAKENHI Grant Number 16K00305. Takimoto is grateful to the supports from JSPS KAKENHI Grant Number 15H02667. In addition, the authors acknowledge the support from MEXT KAKENHI Grant Number 24106010 (the ELC project).

References

1. J. Abernethy, P. L. Bartlett, N. Buchbinder, and I. Stanton. A Regularization Approach to Metrical Task Systems. In *Proceedings of the 21st International Conference on Algorithmic Learning Theory (ALT'10)*, volume LNCS 6331, pages 270–284, 2010.
2. N. Ailon, K. Hatano, and E. Takimoto. Bandit Online Optimization Over the Permutahedron. In *Proceedings of 25th International Conference on Algorithmic Learning Theory(ALT 2014)*, volume 8776 of *LNCS*, pages 215–229, 2014.
3. N. Bansal, N. Buchbinder, A. Madry, and J. S. Naor. A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5):40:1–40:49, Nov. 2015.
4. Y. Bartal, A. Blum, C. Burch, and A. Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 711–719, New York, NY, USA, 1997. ACM.
5. Y. Bartal, B. Bollobas, and M. Mendel. A ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 396–405, Oct 2001.
6. A. Blum, S. Chawla, and A. Kalai. Static optimality and dynamic search-optimality in lists and trees. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 1–8, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.
7. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, New York, NY, USA, 1998.
8. A. Borodin, N. Linial, and M. E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, Oct. 1992.
9. N. Buchbinder, S. Chen, J. S. Naor, and O. Shamir. Unified algorithms for online learning and competitive analysis. *Mathematics of Operations Research*, 41(2):612–625, 2016.
10. N. Cesa-Bianchi and G. Lugosi. Combinaotrial Bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.
11. A. Fiat and M. Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6):1403–1422, 2003.
12. Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
13. A. Gupta, K. Talwar, and U. Wieder. Changing bases: Multistage optimization for matroids and matchings. In *ICALP (1)*, volume 8572 of *Lecture Notes in Computer Science*, pages 563–575. Springer, 2014.
14. S. Irani and S. Seiden. Randomized algorithms for metrical task systems. *Theoretical Computer Science*, 194(12):163 – 182, 1998.
15. M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51:671–697, 2004.
16. A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.
17. E. Koutsoupias and C. H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5):971–983, Sept. 1995.

18. E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4(5):773–818, 2004.

A On Assumption 1

As is well known as a folklore, we can assume without loss of generality that the loss vectors $\boldsymbol{\ell}_t$ are small enough, so that Assumption 1 is satisfied.

Assumption 1 *Whenever the previous state* $\boldsymbol{c}_{t-1}$ *satisfies* $\boldsymbol{c}_{t-1} \cdot \boldsymbol{L} < L$, *where* $\boldsymbol{L}$ *is the cumulative loss vectors up to round* $t-1$ *in the current phase, and the phase did not end at round* $t-1$, *i.e.,* $\min_{\boldsymbol{c}^* \in C} \boldsymbol{c}^* \cdot \boldsymbol{L} < 1$, *then* $\boldsymbol{\ell}_t$ *satisfies the two conditions:*

1. $\boldsymbol{c}_{t-1} \cdot (\boldsymbol{L} + \boldsymbol{\ell}_t) \le L$, *and*
2. $\min_{\boldsymbol{c}^* \in C} \boldsymbol{c}^* \cdot (\boldsymbol{L} + \boldsymbol{\ell}_t) \le 1$.

This is because, when $\boldsymbol{\ell}_t$ violates the assumption, then we can replace $\boldsymbol{\ell}_t$ by a sequence of non-negative loss vectors $\boldsymbol{\ell}_{t_1}, \boldsymbol{\ell}_{t_2}, \ldots, \boldsymbol{\ell}_{t_k}$ so that $\boldsymbol{\ell}_t = \boldsymbol{\ell}_{t_1} + \cdots + \boldsymbol{\ell}_{t_k}$ and the new sequence of loss vectors satisfy the assumption in the following way:

1. If the first condition is violated, i.e., $\boldsymbol{c}_{t-1} \cdot (\boldsymbol{L} + \boldsymbol{\ell}_t) = a > L$, then we let
$$\alpha_1 = \frac{L - \boldsymbol{c}_{t-1} \cdot \boldsymbol{L}}{a - \boldsymbol{c}_{t-1} \cdot \boldsymbol{L}}.$$
Otherwise, we let $\alpha_1 = 1$. In the former case, we can easily verify that $0 < \alpha_1 < 1$ and $\boldsymbol{c}_{t-1} \cdot (\boldsymbol{L} + \alpha_1 \boldsymbol{\ell}_t) = L$.
2. If the second condition is violated, i.e., $\min_{\boldsymbol{c}^* \in C} \boldsymbol{c}^* \cdot (\boldsymbol{L} + \boldsymbol{\ell}_t) > 1$, then we let $0 < \alpha_2 < 1$ be such that $\min_{\boldsymbol{c}^* \in C} \boldsymbol{c}^* \cdot (\boldsymbol{L} + \alpha_2 \boldsymbol{\ell}_t) = 1$. Otherwise, we let $\alpha_2 = 1$. Note that, in the former case, we can find such α_2 efficiently by binary search.
3. Let $\alpha = \min\{\alpha_1, \alpha_2\}$ and $\boldsymbol{\ell}_{t_1} = \alpha \boldsymbol{\ell}_t$ and $\boldsymbol{\ell}_{t_2} = (1-\alpha)\boldsymbol{\ell}_t$. Then, clearly $\boldsymbol{\ell}_{t_1}$ satisfies Assumption 1. If $\boldsymbol{\ell}_{t_2}$ still violates the assumption, then repeat the same procedure for $\boldsymbol{\ell}_{t_2}$ recursively.