Seminar on Online Algorithms valentinpi
Freie Universität Berlin May 2, 2023
Summer Term 2023

Presentation Notes on: Metrical Task Systems and the Work Function Algorithm

We present an abstract model for the competitive analysis of online algorithms, discuss a general solution algorithm and state some bounds for specific problems.

Preliminaries We shall first recall the definition of a metric.

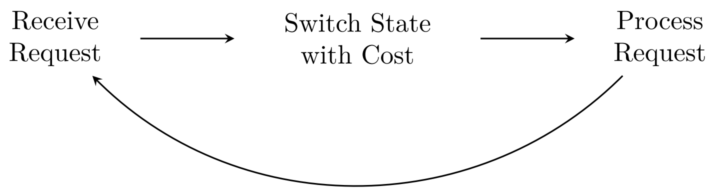
Definition 1 ([1, pp. 3-4])**.** Let $\mathcal{M}$ be a set. A *metric* is a map $d\colon \mathcal{M}\times\mathcal{M} \to \mathbb{R}$ with the following properties for any $x, y, z \in \mathcal{M}$:

(i) $d(x, y) = 0 \leftrightarrow x = y$
(ii) $d(x, y) = d(y, x)$ (Symmetry)
(iii) $d(x, z) \leq d(x, y) + d(y, z)$ (Triangle Inequality)

A tuple $(\mathcal{M}, d)$ is called a *metric space*.

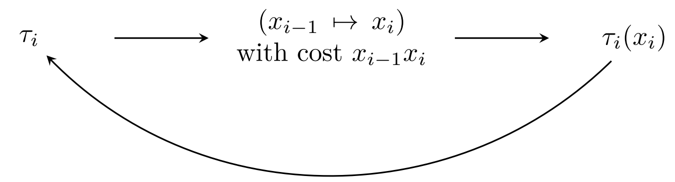
An immediate consequence of this definition is the positivity of d, as it is trivial for an empty metric space and for a nonempty one, we have $0 = d(x, x) \leq d(x, y) + d(y, x) = 2d(x, y)$ for $x, y \in \mathcal{M}$. Furthermore, when denoting a metric space, we usually omit the symbol for the metric and just write $\mathcal{M}$. To simplify the notation, let $xy \coloneqq d(x, y)$. It is important to note that metric spaces abstract the notion of a distance and that these spaces can be finite.

Metrical Task Systems The basic premise in online algorithm design is, that the full input of an algorithm is not fully known in advance. We usually model this circumstance the following way: The input is slowly revealed by the algorithm receiving requests from another party. It is then in some state, which we can think of as the algorithm having prepared some memory for processing the input that is most expected to come up next. The algorithm may then need to switch the state, process the request and iterate this procedure.


$$\begin{tikzcd}
\text{Receive Request} \arrow[r] & \text{Switch State with Cost} \arrow[r] & \text{Process Request} \arrow[ll, bend left]
\end{tikzcd}$$

Definition 2 ([2, pp. 75-76])**.** A *Metrical Task System* (MTS) is a tuple $((\mathcal{M}, d), \mathcal{T})$, where $(\mathcal{M}, d)$ is a finite metric space of cardinality $N \coloneqq |\mathcal{M}|$, the *set of states*, and $\mathcal{T} \subseteq \left(\mathbb{R}^{\infty}_{\geq 0}\right)^N$ is the *set of tasks*. If $d(x, y) = 1$ for any $x, y \in \mathcal{M}$ with $x \neq y$, then the MTS is called *uniform*.

Given a fixed order of the elements in $\mathcal{M}$, any task $\tau \in \mathcal{T}$ can also be interpreted as a map of form $\{0, 1, ..., N - 1\} \to \mathbb{R}^{\infty}_{\geq 0}$, so we may write $\tau(x)$ for $x \in \mathcal{M}$. The term $\tau(x)$ corresponds to the time to process the task τ in the state x. We will now also omit the term $\mathcal{T}$, when specifying an MTS.


$$\begin{tikzcd}
\tau_i \arrow[r] & \begin{array}{c}(x_{i-1} \mapsto x_i) \\ \text{with cost } x_{i-1}x_i\end{array} \arrow[r] & \tau_i(x_i) \arrow[ll, bend left]
\end{tikzcd}$$

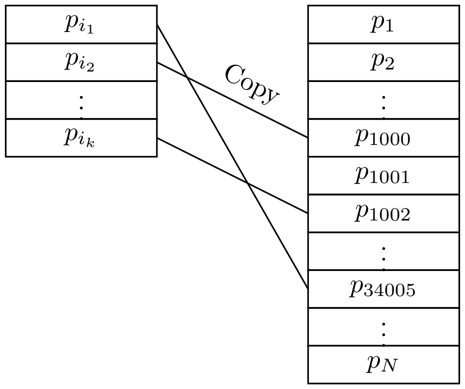
As we can see, the MTS model fits our description of how a system running an online algorithm behaves in theory. To get acquainted with definition 2, we shall describe three examples.

Example 1 (The Ice Cream Problem [2, pp. 74-75]). Suppose we work at an ice cream shop, which offers two flavors, vanilla (V) and strawberry (S). We can manually produce a gallon of vanilla ice cream at a cost of 2 and of strawberry ice cream at a cost of 4. The shop has an ice cream automaton, which can produce both flavors at half the cost each, but it can only produce gallons of one kind at a time (so either vanilla V_M or chocolate S_M) and needs to switch its mode at a cost of 1. The question is when to change the mode of the automaton to keep the cost low while serving customers. The following matrix captures these costs.

	V	S
V_M	1	4
S_M	2	2

The MTS modelling this problem would be composed of the space $\{V_M, S_M\}$ (in this order) of states with the metric $d\colon (V_M, V_M) \mapsto 0, (V_M, S_M) \mapsto 1, (S_M, V_M) \mapsto 1, (S_M, S_M) \mapsto 0$ and the set of tasks $\mathcal{T} := \{(1,4), (2,2)\}$. The MTS is uniform.

Example 2 (The Paging Problem [3, p. 124]). Suppose we have an operating system that utilizes *paging* to implement memory management. We have a large, slow memory of $N \in \mathbb{N}$, $N \geq 1$ pages and a small, fast memory of $k \in \mathbb{N}$, $1 \leq k \leq N$ pages. We can model the possible states of the smaller memory with a metric space $\mathcal{M}$ of $\binom{N}{k}$ elements, representing which k pages are currently in the small memory, assuming that the memory is filled at all times.

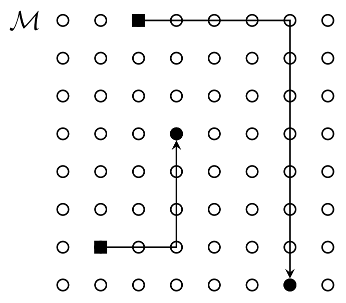


In the figure, $p_1, ..., p_N$ denote the pages and $i_1, ..., i_k \in \mathbb{N}$, $1 \leq i_1, ..., i_k \leq N$ denote pairwise different indices. Denoting the states by $S_1, ..., S_{\binom{N}{k}}$, a suitable metric is $d(S_i, S_j) := k - |S_i \cap S_j|$, for which we can verify the properties of a metric, only the triangle inequality requires a case distinction. We now set $\mathcal{T}$ as an N-set, for which every task τ is associated with a page p. As the task should output the page requested, we let

$$\tau(S_i) := \begin{cases} 0 & p \in S_i \\ \infty & p \notin S_i \end{cases} \tag{1}$$

for any $i \in \mathbb{N}$, $1 \leq i \leq \binom{N}{k}$ to force a suitable online algorithm to switch the state to one which contains the requested page. This MTS is not uniform for $N > 3$ and $k > 1$.

Example 3 (The k-Server Problem [2, pp. 87-88]). We are given a metric space $\mathcal{M}$ and k servers which reside on one point of the space each. A request for the entire system arrives at one of the points in the space, meaning that one of the servers needs to move to the point to serve the request. Wlog. we assume, that two or more servers can sit at the same point.
A suitable formulation as an MTS would be to equip $\mathcal{M}^k$ with the metric assigning two configurations the minimum distance the servers need to travel from one configuration starting to cover the same points in the same weights as in the other configuration, also called the *configuration distance* of two *configurations*, i.e. states. To each task τ we then have a point $x \in \mathcal{M}$ associated, s.t. $\tau(x_1, ..., x_k) = 0$, iff $x = x_i$ for an $i \in \mathbb{N}$, $1 \leq i \leq k$. Otherwise $\tau(x_1, ..., x_k) = \infty$.



Online Algorithms With the definition of an MTS, we can now especially formulate the concept of an online algorithm in the MTS model, as well as the optimal algorithm and competitiveness. For the following paragraphs, let $\mathcal{M}$ denote an MTS, $x_0 \in \mathcal{M}$ an initial state and $\tau \in \mathcal{T}^n$, $n \in \mathbb{N}$, be a one-indexed sequence of tasks.

Definition 3 ([2, p. 76])**.** Let $x \in \mathcal{M}^n$ be a one-indexed sequence of states. Set

$$\text{cost}(x_0, \tau, x) := \sum_{i=1}^{n} x_{i-1}x_i + \tau_i(x_i) \tag{2}$$

$$\text{opt}(x_0, \tau) := \min_{x' \in \mathcal{M}^n} \text{cost}(x_0, \tau, x') \tag{3}$$

as the cost of processing the task sequence τ under the state sequence x and the optimal cost respectively. Let further $\mathcal{T}^* := \bigcup_{m=1}^{\infty} \mathcal{T}^m$. A (deterministic) *online strategy/online algorithm* for $\mathcal{M}$ is a map $\mathcal{A}\colon \mathcal{M} \times \mathcal{T}^* \to \mathcal{M}$. We further set the cost of execution by $\mathcal{A}$ as

$$\text{cost}_{\mathcal{A}}(x_0, \tau) := \text{cost}(x_0, \tau, \mathcal{A}(x_0, \tau_1)\mathcal{A}(x_0, \tau_1\tau_2)...\mathcal{A}(x_0, \tau_1\tau_2...\tau_n)) \tag{4}$$

$\mathcal{A}$ is called *c-competitive with initial function* $\alpha\colon \mathcal{M} \to \mathcal{R}$, where $c \in \mathbb{R}$, if for any $x_0 \in \mathcal{M}$ and $\tau \in \mathcal{T}^*$, we have

$$\text{cost}_{\mathcal{A}}(x_0, \tau) \leq c \,\text{opt}(x_0, \tau) + \alpha(x_0) \tag{5}$$

Furthermore, the value $\text{argmin}_{c \in \mathbb{R}}(\mathcal{A}$ is c-competitive), if it exists, is called the *competitive-ratio* of $\mathcal{A}$.

The definitions from the model correspond to the same definition of competitiveness as discussed in class, which hints at the abstractions made being well-chosen. As mentioned, this is the definition for deterministic online algorithms. At the end, we will mention a few different models.

The Work Function Algorithm In this part, we will give an optimal competitive algorithm for an arbitrary MTS with the competitive ratio only depending on N, the so-called *Work Function Algorithm* (WFA). We first need to define what *work functions* are.

Definition 4 (Work Functions)**.** The function

$$\omega\colon \mathcal{M} \to \mathbb{R}, x \mapsto \min_{(x_1,...,x_{n-1}) \in \mathcal{M}^{n-1}} \text{cost}(x_0, \tau, (x_0, x_1, ..., x_{n-1}, x)) \tag{6}$$

is called the *work function* for x_0 and τ.

The work function describes the minimal cost for reaching a given state x after processing τ. The work function can be computed using a dynamical program. Let ω_i for any $i \in \mathbb{N}$, $0 \leq i \leq n$, denote the work function of x_0 and $(\tau_1, ..., \tau_i)$ computing the lowest costs for the first i tasks. Then

$$\omega_0(x) = x_0 x \tag{7}$$

$$\omega_{i+1}(x) = \min_{x' \in \mathcal{M}} \omega_i(x') + \tau_{i+1}(x') + x'x \tag{8}$$

We can verify the correctness through induction: The base formula for ω_0 is clear. Then the lowest cost for computing the value of ω_{i+1} is given by the above term and is also optimal by the induction assumption. The work function and our ability to compute it now allows the formulation of the so-called *Work Function Algorithm*. In the same manner as the formal definition we describe it as the function WFA: $\mathcal{M} \times \mathcal{T}^* \to \mathcal{M}$.

The Work Function Algorithm (WFA) Let $i \in \mathbb{N}$, $1 \leq i \leq n-1$ and suppose the states $x_0, x_1, ..., x_i$ have already been computed. Then the WFA chooses the next state by

$$x_{i+1} \in \arg\min_{x \in \mathcal{M}} \omega_{i+1}(x) + x_i x \tag{†}$$

$$\text{s.t. } \omega_{i+1}(x_{i+1}) = \omega_i(x_{i+1}) + \tau_{i+1}(x_{i+1}) \tag{‡}$$

This algorithm was first suggested by Ricklin, based on results by Borodin, Linial and Saks [3, p. 123, p. 132]. A rough interpretation may be, that the algorithm chooses the next state by minimizing the cost to process $(\tau_1, ..., \tau_{i+1})$ and to move from x_i to x_{i+1}. The second condition also establishes, that in the state x_{i+1}, we process τ_{i+1}. It is not clear why it is optimal, nor why it is well-defined, which is the first question we are going to tackle.

Theorem 4 ([3, pp. 132-133])**.** The WFA is well-defined.

Proof. As $\mathcal{M}$ is finite, there is always an element sufficing (†). We first prove the direction ($\leq$) for any such element and then prove the existence of one sufficing ($\geq$) for (‡).
($\leq$) Notice, that for any $x \in \mathcal{M}$, we have from the definition of the work function, that

$$\underbrace{\omega_{i+1}(x)}_{\substack{\text{Optimal cost for processing} \\ (\tau_1, ..., \tau_{i+1}) \text{ and ending in } x.}} \leq \underbrace{\omega_i(x) + \tau_{i+1}(x)}_{\substack{\text{Optimal cost for processing } (\tau_1, ..., \tau_i), \\ \text{ending in } x \text{ and then processing } \tau_{i+1} \text{ there.}}} \tag{9}$$

($\geq$) Let

$$x' \in \arg\min_{x \in \mathcal{M}} \omega_{i+1}(x) + x_i x \tag{10}$$

$$x^* \in \arg\min_{x \in \mathcal{M}} \omega_i(x) + \tau_{i+1}(x) + x x' \tag{11}$$

These elements give the minimas from (†) and (8), where the latter is applied on x'. We claim

$$\omega_{i+1}(x^*) + x_i x^* \leq \omega_{i+1}(x') + x_i x' \tag{12}$$

For that, we have the three inequalities

$$\omega_{i+1}(x^*) + x^* x_i \leq \omega_i(x^*) + \tau_{i+1}(x^*) + x^* x_i \tag{13}$$

$$\omega_i(x^*) + \tau_{i+1}(x^*) + x^* x_i = \omega_{i+1}(x') + x^* x_i - x^* x' \tag{14}$$

$$x^* x_i - x^* x' \leq x' x_i \tag{15}$$

We obtain (13) by taking (9) for $x = x^*$ and adding $x^* x_i$. (14) is obtained by taking the statement in (11) and adding $x^* x_i - x^* x'$. (15) is then obtained by the triangle inequality of the metric. Combining all three statements by following the equalities and inequalities from left to right, top to bottom, we obtain the statement. By that, we first have that x^* satisfies (†), as x' minimizes the term involved, additionally giving that our claim is an equality. ($\geq$) now follows from

$$\omega_{i+1}(x^*) + x^* x_i \overset{(1)}{=} \omega_{i+1}(x') + x' x_i \overset{(2)}{=} \omega_i(x^*) + \tau_{i+1}(x^*) + x' x^* + x' x_i \overset{(3)}{\leq} \omega_i(x^*) + \tau_{i+1}(x^*) + x^* x_i \tag{16}$$

(1) Using that (12) is an equality.
(2) Due to (11).
(3) By the triangle inequality.

giving the statement. ■

We now analyze the competitiveness of the WFA. For that, we will require the following lemma.

Theorem 5 ([3, pp. 133-134])**.** For any $i \in \mathbb{N}$, $0 \leq i \leq n$, we have

$$\text{cost}_{\text{WFA}}(x_0, (\tau_1, ..., \tau_i)) + B_0 \leq B_i \leq (2N-1)\, \text{opt}(x_0, (\tau_1, ..., \tau_i)) + (2N-2)\mu \tag{17}$$

where $(x_1, ..., x_n) \in \mathcal{M}^n$ is the sequence of states produced by the WFA and

$$\mu := \max_{x,x' \in \mathcal{M}} xx',\ B_i := 2 \sum_{x \in \mathcal{M} \setminus \{x_i\}} \omega_i(x) + \omega_i(x_i) \tag{18}$$

In other words, the WFA is $(2N-1)$-competitive.

Proof. We start by proving the first inequality. By (†),

$$\omega_{i+1}(x_{i+1}) + x_i x_{i+1} = \min_{x \in \mathcal{M}} \omega_{i+1}(x) + x_i x \leq \omega_{i+1}(x_i) + x_i x_i = \omega_{i+1}(x_i) \tag{19}$$

And by (‡), $\omega_{i+1}(x_{i+1}) - \omega_i(x_{i+1}) = \tau_{i+1}(x_{i+1})$. Reordering the first result and plugging in the second, we have

$$\omega_{i+1}(x_i) - \omega_i(x_{i+1}) \geq x_i x_{i+1} + \tau_{i+1}(x_{i+1}) \tag{20}$$

We claim $B_{i+1} - B_i \geq \omega_{i+1}(x_i) - \omega_i(x_{i+1})$. First, notice that ω_i is monotonely increasing wrt. i, as by (8), we have for any $x \in \mathcal{M}$

$$\omega_{i+1}(x) = \min_{x' \in S} \omega_i(x') + \tau_{i+1}(x') + x'x \geq \omega_i(x) \tag{21}$$

So

$$B_{i+1} - B_i = 2 \left(\sum_{x \in \mathcal{M} \setminus \{x_i, x_{i+1}\}} \underbrace{\omega_{i+1}(x) - \omega_i(x)}_{\geq 0} \right) \tag{22}$$

$$+ \underbrace{\omega_{i+1}(x_{i+1}) - \omega_i(x_{i+1})}_{\geq 0} + \underbrace{\omega_{i+1}(x_i) - \omega_i(x_i)}_{\geq 0} + (\omega_{i+1}(x_i) - \omega_i(x_{i+1}))) \tag{23}$$

This gives the claim. Especially, we now have $B_{i+1} - B_i \geq x_i x_{i+1} + \tau_{i+1}(x_{i+1})$, giving the telescope sum

$$\text{cost}_{\text{WFA}}(x_0, (\tau_1, ..., \tau_i)) = \sum_{j=1}^{i} x_{j-1} x_j + \tau_j(x_j) \leq \sum_{j=1}^{i} B_j - B_{j-1} = B_i - B_0 \tag{24}$$

and by that giving the first inequality.
For the second inequality, observe from the definition, that $\omega_i(x) = \min_{x' \in \mathcal{M}} \omega_i(x') + x'x \leq \min_{x' \in \mathcal{M}} \omega_i(x') + \mu$. There is further at least one end state giving the optimum cost, s.t. plugging that in gives

$$B_i = 2 \sum_{x \in \mathcal{M} \setminus \{x_i\}} \omega_i(x) + \omega_i(x_i) \leq (2N-2)(\text{opt}(x_0, (\tau_1, ..., \tau_i)) + \mu) + \text{opt}(x_0, (\tau_1, ..., \tau_i)) \tag{25}$$

■

Lastly, we can establish that $(2N-1)$ is also the competitive ratio of the WFA, but we are going to omit that here. The following theorem only presents this result for general deterministic online algorithms.

Theorem 6 ([3, pp. 128-132])**.** The competitive ratio of any deterministic online algorithm for general MTS is lower bounded by $(2N-1)$.

One last remark: As we can observe, the WFA runs in time $O(nN^2)$. A runtime of $O(nN^2)$ is required to compute the first n work functions, for which the algorithm needs to compute N values each taking minimums over the N states. Each of the n algorithm states then requires checking all N values of a work function.

Results on WFAs for Some Problems The following table gives an overview of results on the initial examples, as well as the different versions of the k-server problem. The values c_{WFA} and c_{Better} shall denote the competitiveness of the associated WFA and some better known algorithms. We know the first two examples already. Especially for the k-server problem however, there are multiple results on different restrictions of the problem, illustrating the richness of that problem. Especially, the following conjecture, still open after much effort, exist.

Conjecture 1 ([3, pp. 152-153])**.** There exists a k-competitive deterministic online algorithm for the k-server algorithm over any metric space.

Problem	c_{WFA}	c_{Better}
Ice Cream	3	7/6 (claim in exercise, [2, pp. 80-82])
Paging	$2\binom{N}{k} - 1$	k (tight, [2, pp. 54-56])
k-Server MTS WFA	$2N^k - 1$	See below
2-Server WFA	2 (tight, [2, pp. 87-89, p. 94])	None
k-Server WFA, $k \geq 3$	$2k - 1$ ([2, pp. 92-93])	Open
k-Server WFA, $\lvert\mathcal{M}\rvert = k + 2$	k (tight, [2, pp. 87-89, pp. 94-95])	None

Especially the example of paging shows how weak the WFA can be in comparison to other online algorithms, but the generality still allows to prove the existence of a competitive algorithm for an any online algorithmic problem, no matter how difficult, if it can be expressed as an MTS. Especially, in the case of algorithms for the k-server problem, it can lead to conjectured nearly optimal algorithms.

References

[1] O. Forster, *Analysis 2*, ISBN: 978-3-658-19410-9.
[2] G. Woeginger, *Online Algorithms*, ISBN: 978-3-540-64917-4.
[3] A. Borodin, *Online Computation and Competitive Analysis*, ISBN: 978-0-521-56392-5.