Online Metric Algorithms with Untrusted Predictions

Antonios Antoniadis\textsuperscript{1}  Christian Coester\textsuperscript{2}  Marek Elias\textsuperscript{3}  
Adam Polak\textsuperscript{4}  Bertrand Simon\textsuperscript{5}

1: MPI, Saarbrucken (Germany).
2: CWI, Amsterdam (Netherlands).
3: EPFL, Lausanne (Switzerland).
4: Jagiellonian University, Kraków (Poland).
5: University of Bremen (Germany).

Dagstuhl – February 2020
Motivation

Online algorithms

▶ Guaranteed competitive ratio
▶ Bad performance on easy instances, overly pessimistic

Machine Learning predictions

▶ Often relevant information
▶ No guarantee, can be arbitrarily bad

Prediction-augmented algorithms

▶ Target competitive ratio: $O\left( \min\{ 1 + f(\eta/OPT), \ ONLINE \} \right)$
Motivation

Online algorithms

- Guaranteed competitive ratio
- Bad performance on easy instances, overly pessimistic

Machine Learning predictions

- Often relevant information
- No guarantee, can be arbitrarily bad

Prediction-augmented algorithms

- Target competitive ratio: $O(\min\{1 + f(\eta/OPT), ONLINE\})$
Motivation

**Online algorithms**
- Guaranteed competitive ratio
- Bad performance on easy instances, overly pessimistic

**Machine Learning predictions**
- Often relevant information
- No guarantee, can be arbitrarily bad

Prediction-augmented algorithms
- Target competitive ratio: $O(\min\{1 + f(\eta/OPT), \text{ONLINE}\})$
Some previously studied problems

- Ski rental: predict #days we will ski [KumarPS’18]
- Non-clairvoyant scheduling: predict processing times [KumarPS’18]
- Restricted assignment: predict machine weights [LattanziMLV’20]
- Caching: predict next arrival time [LykourisV’18, Rohatgi’20]

Issue: lack of generality, predictions tailored to specific problems

Lemma

Previously used caching predictions are not useful for weighted caching.

→ need for a more general prediction setup
The Metrical Task System (MTS) problem

Definition by picture

Round 0
Cost incurred: 0

Note: generalizes caching, $k$-server, convex body chasing...

Prediction setup

▸ Fix OFF: offline algorithm (e.g., OPT), who goes to states $o_1, o_2, \ldots$

▸ At time $t$, $p_t := \text{prediction of } o_t$. Error: $\eta = \sum_t d(o_t, p_t)$
The Metrical Task System (MTS) problem

Definition by picture

![Graph representation of the MTS problem]

Round 1 before serving
Cost incurred: 0

Note: generalizes caching, \( k \)-server, convex body chasing...

Prediction setup

- **Fix OFF**: offline algorithm (e.g., \( \text{OPT} \)), who goes to states \( o_1, o_2, \ldots \)
- **At time** \( t \), \( p_t := \text{prediction of } o_t \).

Error: \( \eta = \sum_t d(o_t, p_t) \)
The Metrical Task System (MTS) problem

Definition by picture

Round 1 after serving
Cost incurred: 3+1

Note: generalizes caching, k-server, convex body chasing...

Prediction setup

▶ Fix OFF: offline algorithm (e.g., OPT), who goes to states $o_1, o_2, \ldots$
▶ At time $t$, $p_t :=$ prediction of $o_t$. Error: $\eta = \sum_t d(o_t, p_t)$
The Metrical Task System (MTS) problem

**Definition by picture**

Round 2 before serving  
Cost incurred: 3+1

Note: generalizes caching, \( k \)-server, convex body chasing...

**Prediction setup**

- **Fix OFF**: offline algorithm (e.g., OPT), who goes to states \( o_1, o_2, \ldots \)
- **At time** \( t \), \( p_t := \) prediction of \( o_t \).

Error:  
\[ \eta = \sum_t d(o_t, p_t) \]
The Metrical Task System (MTS) problem

**Definition by picture**

Round 2 after serving
Cost incurred: 3+1+7+2

Note: generalizes caching, \( k \)-server, convex body chasing...

**Prediction setup**

- **Fix OFF**: offline algorithm (e.g., OPT), who goes to states \( o_1, o_2, \ldots \)
- **At time** \( t \), \( p_t := \text{prediction of } o_t \).

Error: \( \eta = \sum_t d(o_t, p_t) \)
Algorithm for MTS: \texttt{ROBUSTFtP}

\textbf{Algorithm subroutine (FtP, Follow the Prediction)}

- Go to $p_t$, except if there is a cheap state nearby
- Proposition: this costs at most $\text{OFF} \cdot (1 + 4\eta/\text{OFF})$

\textbf{Combining online algorithms A and B: $\text{comb}(A, B)$ \textit{[BlumB’00]}}

- $E(\text{cost}_{\text{comb}(A,B)}) \leq (1 + \varepsilon) \cdot \min\{E(\text{cost}_A), E(\text{cost}_B)\}$ asymptotically

\textbf{Theorem (ROBUSTFtP := $\text{comb}(\text{ONLINE}, \text{FtP})$ competitive ratio)}

\texttt{ROBUSTFtP} is $O(\min\{1 + \eta/\text{OFF} , \text{ONLINE}\})$ - competitive.

(Recall: $\eta = \sum_t d(o_t, p_t)$)

\textbf{Lemma (Lower bound)}

This is tight for some MTS (i.e., $\eta/\text{OFF}$ - dependency).
Logarithmic error dependency for caching

Issue with \textsc{RobustFtP}: too general, not the best for all MTS

**Focus on a specific MTS: the caching problem**

- Maintain a cache of \(k\) pages, pay 1 per cache miss

**Algorithm \textsc{Trust&Doubt}: explaining how it works its name**

- “Trust” predictor: evict a page not in its cache
- If an evicted page requested: “doubt” this decision
- Regularly (depending on trustworthiness): “trust” again

**Theorem (\textsc{Trust&Doubt} competitive ratio)**

\textsc{Trust&Doubt} costs at most \(O(\text{Off} \cdot \min\{1 + \log \frac{n}{\text{Off}}, \log k\})\).
Comparison to previous caching algorithms

How do our algorithms compare to previous ones?

- Different errors → difficult to compare competitive ratios
- Experimentally: compute previous predictions
deduct our predictions (evict furthest predicted)

Results on a public dataset (BrightKite, \( k = 10 \)) – (lower is better)

<table>
<thead>
<tr>
<th></th>
<th>LRU</th>
<th>Marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLECO</td>
<td>2.081</td>
<td>1.707</td>
</tr>
<tr>
<td>POPU</td>
<td>1.340</td>
<td>1.262</td>
</tr>
<tr>
<td>FT-P</td>
<td>1.337</td>
<td>1.264</td>
</tr>
<tr>
<td>L&amp;V [LykourisV’18]</td>
<td>1.339</td>
<td>1.292</td>
</tr>
<tr>
<td>LMarker [Rohatgi’20]</td>
<td>1.337</td>
<td>1.292</td>
</tr>
<tr>
<td>LNonMarker [Rohatgi’20]</td>
<td>1.339</td>
<td>1.292</td>
</tr>
<tr>
<td>RobustFP</td>
<td>1.351</td>
<td>1.316</td>
</tr>
<tr>
<td>Trust&amp;Doubt</td>
<td>1.292</td>
<td>1.274</td>
</tr>
</tbody>
</table>

Prediction: ground truth + lognorm error

Two predictors: simple statistics
Comparison to previous caching algorithms

How do our algorithms compare to previous ones?

- Different errors → difficult to compare competitive ratios
- Experimentally: compute previous predictions
deduction our predictions (evict furthest predicted)

Results on a public dataset (BrightKite, $k = 10$) – (lower is better)

<table>
<thead>
<tr>
<th></th>
<th>PLECO</th>
<th>POPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>1.291</td>
<td></td>
</tr>
<tr>
<td>Marker</td>
<td>1.333</td>
<td></td>
</tr>
<tr>
<td>Predictions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FtP</td>
<td>2.081</td>
<td>1.707</td>
</tr>
<tr>
<td>L&amp;V \cite{LykourisV'18}</td>
<td>1.340</td>
<td>1.262</td>
</tr>
<tr>
<td>LMarker \cite{Rohatgi'20}</td>
<td>1.337</td>
<td>1.264</td>
</tr>
<tr>
<td>LNonMarker \cite{Rohatgi'20}</td>
<td>1.339</td>
<td>1.292</td>
</tr>
<tr>
<td>RobustFtP</td>
<td>1.351</td>
<td>1.316</td>
</tr>
<tr>
<td>Trust&amp;Doubt</td>
<td>1.292</td>
<td>1.274</td>
</tr>
</tbody>
</table>

Two predictors: simple statistics

Prediction: ground truth + lognorm error
Comparison to previous caching algorithms

How do our algorithms compare to previous ones?

- Different errors → difficult to compare competitive ratios
- Experimentally: compute previous predictions
deduct our predictions (evict furthest predicted)

Results on a public dataset (BrightKite, $k = 10$) – (lower is better)

<table>
<thead>
<tr>
<th></th>
<th>LRU</th>
<th>Marker</th>
<th>Predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>PLECO</td>
</tr>
<tr>
<td>FtP</td>
<td>1.291</td>
<td>1.333</td>
<td>2.081</td>
</tr>
<tr>
<td>LMarker [Rohatgi’20]</td>
<td>1.337</td>
<td>1.264</td>
<td>1.337</td>
</tr>
<tr>
<td>RobustFtP</td>
<td>1.351</td>
<td>1.316</td>
<td></td>
</tr>
<tr>
<td>Trust&amp;Doubt</td>
<td>1.292</td>
<td>1.274</td>
<td></td>
</tr>
</tbody>
</table>

Two predictors: simple statistics

Prediction: ground truth + lognorm error

Noise parameter $\sigma$ of the synthetic predictor
Comparison to previous caching algorithms

How do our algorithms compare to previous ones?

- Different errors → difficult to compare competitive ratios
- Experimentally: compute previous predictions
deduct our predictions (evict furthest predicted)

Results on a public dataset (BrightKite, $k = 10$) — *(lower is better)*

<table>
<thead>
<tr>
<th>Prediction: ground truth + lognorm error</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>LRU</th>
<th>Marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pleco</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POPU</td>
<td>1.291</td>
<td>1.333</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predictors</th>
<th>Pleco</th>
<th>POPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP</td>
<td>2.081</td>
<td>1.707</td>
</tr>
<tr>
<td>L&amp;V [LykourisV’18]</td>
<td>1.340</td>
<td>1.262</td>
</tr>
<tr>
<td>L Marker [Rohatgi’20]</td>
<td>1.337</td>
<td>1.264</td>
</tr>
<tr>
<td>L Non Marker [Rohatgi’20]</td>
<td>1.339</td>
<td>1.292</td>
</tr>
<tr>
<td>Robust FTP</td>
<td>1.351</td>
<td>1.316</td>
</tr>
<tr>
<td>Trust &amp; Doubt</td>
<td>1.292</td>
<td>1.274</td>
</tr>
</tbody>
</table>

Two predictors: simple statistics
MTS (and beyond): prediction setup and “optimal” algorithm
Caching: specific algorithm good in theory & practice
Perspective: other MTS (weighted caching, convex body chasing, $k$-server)
is useful!