Online Metric Algorithms with Untrusted Predictions

Antonios Antoniadis[1] Christian Coester[2] Marek Elias[2]
Adam Polak[3] **Bertrand Simon**[4]

1: University of Cologne (Germany).
2: CWI, Amsterdam (Netherlands).
3: Jagiellonian University, Kraków (Poland).
4: CNRS CC-IN2P3, Villeurbanne (France).

Motivation

Online algorithms

- Optimization with incomplete information
- Guaranteed competitive ratio: r s.t. $\forall I\colon \textsc{Alg}(I) \leq r \cdot \textsc{Opt}(I)$
- Bad performance on easy instances, overly pessimistic

Motivation

Online algorithms

- Optimization with incomplete information
- Guaranteed competitive ratio: r s.t. $\forall I\colon \textsc{Alg}(I) \le r \cdot \textsc{Opt}(I)$
- Bad performance on easy instances, overly pessimistic

Motivation

Online algorithms

- Optimization with incomplete information
- Guaranteed competitive ratio: r s.t. $\forall I\colon \textsc{Alg}(I) \leq r \cdot \textsc{Opt}(I)$
- Bad performance on easy instances, overly pessimistic

Machine learning predictions

- Often very powerful

- No guarantee, can be arbitrarily bad

Motivation

Online algorithms

- Optimization with incomplete information
- Guaranteed competitive ratio: r s.t. $\forall I\colon \textsc{Alg}(I) \le r \cdot \textsc{Opt}(I)$
- Bad performance on easy instances, overly pessimistic

Machine learning predictions

- Often very powerful

- No guarantee, can be arbitrarily bad
- prediction error η

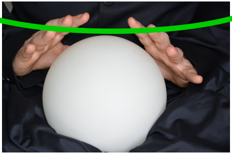
Motivation

Online algorithms

- Optimization with incomplete information
- Guaranteed competitive ratio: r s.t. $\forall I\colon \textsc{Alg}(I) \le r \cdot \textsc{Opt}(I)$
- Bad performance on easy instances, overly pessimistic

Machine learning predictions

- Often very powerful



- No guarantee, can be arbitrarily bad
- prediction error η

Prediction-augmented algorithms

- Target competitive ratio: $O(\ \min\{ONLINE,\ f(\eta/\textsc{Opt})\}\ )$

Some previously studied problems

- Ski rental: predict #days we will ski [PurohitSK'18]
- Non-clairvoyant scheduling: predict processing times [PurohitSK'18]
- Restricted assignment: predict machine weights [LattanziLMV'20]
- Caching: predict next arrival time [LykourisV'18,Rohatgi'20,Wei'20]
- Weighted caching: predict *all* requests until next arrival [JiangPS'20]

Issue: lack of generality, predictions tailored to specific problems

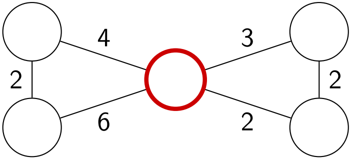
Proposition (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

Previously used caching predictions not useful for weighted caching.

$\Longrightarrow$ need *more general* prediction setup

Metrical Task Systems (MTS)

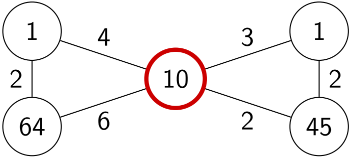
- Given: metric space (M, d), $x_0 \in M$
- At time t:
  1. cost function $\ell_t : M \mapsto \mathbb{R}_+$ revealed
  2. algo chooses $x_t \in M$
  3. pays $d(x_{t-1}, x_t) + \ell_t(x_t)$



Cost: 0

MTS generalizes caching, k-server, convex body/function chasing, layered graph traversal...

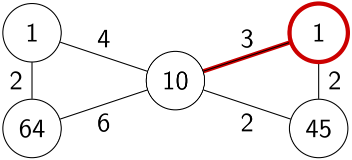
- Given: metric space (M, d), $x_0 \in M$
- At time t:
  1. cost function $\ell_t : M \mapsto \mathbb{R}_+$ revealed
  2. algo chooses $x_t \in M$
  3. pays $d(x_{t-1}, x_t) + \ell_t(x_t)$



Cost: 0

MTS generalizes caching, k-server, convex body/function chasing, layered graph traversal...

- Given: metric space (M, d), $x_0 \in M$
- At time t:
  1. cost function $\ell_t : M \mapsto \mathbb{R}_+$ revealed
  2. algo chooses $x_t \in M$
  3. pays $d(x_{t-1}, x_t) + \ell_t(x_t)$

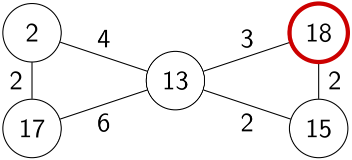


Cost: 3+1

MTS generalizes caching, k-server, convex body/function chasing, layered graph traversal...

Metrical Task Systems (MTS)

- Given: metric space (M, d), $x_0 \in M$
- At time t:
  1. cost function $\ell_t : M \mapsto \mathbb{R}_+$ revealed
  2. algo chooses $x_t \in M$
  3. pays $d(x_{t-1}, x_t) + \ell_t(x_t)$

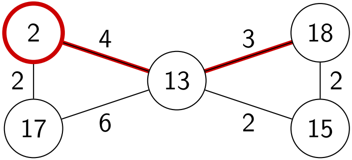


Cost: 3+1

MTS generalizes caching, *k*-server, convex body/function chasing, layered graph traversal...

Metrical Task Systems (MTS)

- Given: metric space (M, d), $x_0 \in M$
- At time t:
  1. cost function $\ell_t : M \mapsto \mathbb{R}_+$ revealed
  2. algo chooses $x_t \in M$
  3. pays $d(x_{t-1}, x_t) + \ell_t(x_t)$

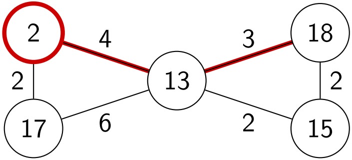


Cost: 3+1+7+2

MTS generalizes caching, k-server, convex body/function chasing, layered graph traversal...

Metrical Task Systems (MTS)

- Given: metric space (M, d), $x_0 \in M$
- At time t:
  1. cost function $\ell_t : M \mapsto \mathbb{R}_+$ revealed
  2. algo chooses $x_t \in M$
  3. pays $d(x_{t-1}, x_t) + \ell_t(x_t)$



Cost: 3+1+7+2

MTS generalizes caching, k-server, convex body/function chasing, layered graph traversal...

What to predict?

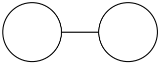
- Next costs?
  - only ℓ_{t+1}? useless with dummy rounds
  - entire future? too much information
- **Single state per round**: recommendation for x_t

Definition of the error and a simple algorithm

Definition: error η

- Fix $\textsc{Off}$: offline algorithm (e.g., $\textsc{Opt}$), who goes to states $o_1, o_2, \dots$
- At time t, $p_t :=$ prediction of o_t.
- Error: $\eta := \sum_t d(o_t, p_t)$

Goal: Small cost (rel. to $\textsc{Off}$) if η small



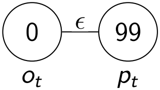
Definition of the error and a simple algorithm

Definition: error η

- Fix $\textsc{Off}$: offline algorithm (e.g., $\textsc{Opt}$), who goes to states $o_1, o_2, \ldots$
- At time t, $p_t :=$ prediction of o_t.
- Error: $\eta := \sum_t d(o_t, p_t)$

Goal: Small cost (rel. to $\textsc{Off}$) if η small

- Naive algo: $x_t := p_t$



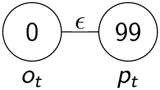
Definition of the error and a simple algorithm

Definition: error η

- Fix $\textsc{Off}$: offline algorithm (e.g., $\textsc{Opt}$), who goes to states $o_1, o_2, \ldots$
- At time t, $p_t :=$ prediction of o_t.
- Error: $\eta := \sum_t d(o_t, p_t)$

Goal: Small cost (rel. to $\textsc{Off}$) if η small

- Naive algo: $x_t := p_t$
- **Better:** $x_t := \arg\min_{x \in X}\{cost_t(x) + 2d(p_t, x)\}$
- Call this algo $\textsc{FtP}$ (Follow the Prediction)



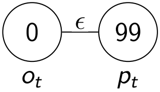
Definition of the error and a simple algorithm

Definition: error η

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

Goal: Small cost (rel. to Off) if η small

- Naive algo: $x_t := p_t$
- **Better:** $x_t := \arg\min_{x \in X}\{cost_t(x) + 2d(p_t, x)\}$
- Call this algo FtP (Follow the Prediction)



Lemma (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

FtP *has cost* $\leq$ Off $+ 4\eta$. *So competitive ratio vs* Off *is* $1 + 4\eta/\text{Off}$.

Guarantee holds simultaneously for **all** offline algos (η depending on it)

Issue: FtP not robust

Robustifying FTP

Combine online algorithms A and B: *comb*(A, B) **[BlumB'00]**

- $E(cost_{comb(A,B)}) \leq (1+\varepsilon)\cdot\min\{E(cost_A), E(cost_B)\} + O(\text{diameter}/\varepsilon)$

Let ROBUSTFTP := $comb(ONLINE, \text{FTP})$

Robustifying FTP

Combine online algorithms A and B: *comb*(A, B) **[BlumB'00]**

- $E(cost_{comb(A,B)}) \leq (1+\varepsilon) \cdot \min\{E(cost_A), E(cost_B)\} + O(\text{diameter}/\varepsilon)$

Let $\textsc{RobustFtP} := comb(ONLINE, \textsc{FtP})$

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$ *has cost* $O(\min\{\textsc{Off} + \eta,\ ONLINE\})$.

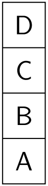
(Recall: η = prediction error wrt. $\textsc{Off}$)

This is asymptotically optimal.

- Lower bound holds for some MTS
- For caching (a special case of MTS), we can do better

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

$k = 4$

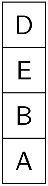


E

misses: 1

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

$k = 4$



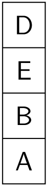
E A

misses: 1

Caching (aka Paging)

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

$k = 4$



E A F

misses: 2

Caching (aka Paging)

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

$k = 4$

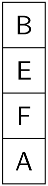
D
E
F
A

E A F B

misses: 3

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

$k = 4$



E A F B E

misses: 3

Caching (aka Paging)

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

$k = 4$

B
E
F
A

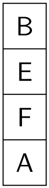
E A F B E B

misses: 3

Caching (aka Paging)

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

$k = 4$



E A F B E B C

misses: 4

Caching (aka Paging)

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

$k = 4$

B
E
F
C

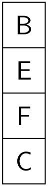
E A F B E B C F

misses: 4

Caching (aka Paging)

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

$k = 4$



E A F B E B C F

misses: 4

- Predicted state: set of k pages

  (intuition: recommended cache content)

Logarithmic error dependence for caching

Algorithm TRUST&DOUBT: idea

- Balance 2 competing policies:
  - "Trust": Evict what is evicted from predicted cache
  - "Doubt": Evict according to classical policy
- In a phase, first follow "Trust"
- If this turns out bad, switch to "Doubt"
- Regularly (depending on trustworthiness): "Trust" again

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

TRUST&DOUBT *is* $\min\left\{O(\log k), O(\log \frac{\eta}{\text{OPT}})\right\}$*-competitive.*

Logarithmic error dependence for caching

Algorithm TRUST&DOUBT: idea

- Balance 2 competing policies:
  - "Trust": Evict what is evicted from predicted cache
  - "Doubt": Evict according to classical policy
- In a phase, first follow "Trust"
- If this turns out bad, switch to "Doubt"
- Regularly (depending on trustworthiness): "Trust" again

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

TRUST&DOUBT *is* $\min\left\{O(\log k), O(\log \frac{\eta}{\text{OPT}})\right\}$*-competitive.*

Logarithmic error dependence for caching

Algorithm TRUST&DOUBT: idea

- Balance 2 competing policies:
  - "Trust": Evict what is evicted from predicted cache
  - "Doubt": Evict according to classical policy
- In a phase, first follow "Trust"
- If this turns out bad, switch to "Doubt"
- Regularly (depending on trustworthiness): "Trust" again

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

TRUST&DOUBT *is* $\min\left\{O(\log k), O(\log \frac{\eta}{\text{OFF}})\right\}$*-competitive vs* OFF.

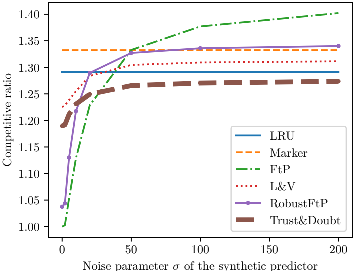
Comparison with previous prediction setup for caching

- Our setup: Prediction = recommended cache content
- In [LykourisV'18, Rohatgi'20, Wei'20]: Prediction = next time of each request

How do setups/results compare?

- Can convert request time predictions to cache content predictions
  - Predictor evicts furthest predicted page
- Prediction error η incomparable
  - Nonetheless: Trust&Doubt also achieves guarantee of [Wei'20]
- Succinct: Instead of $\log T$ bit predictions per request, only $\log k$ bits when predictor has cache miss
- Our setup generalizes to MTS

Same datasets as [LykourisV'18]



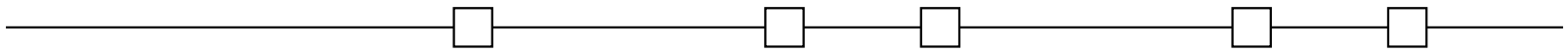
Prediction: ground truth + lognorm error

Dataset	BK		Citi	
LRU	1.29		1.85	
Marker	1.33		1.86	
Predictions	PLECO	POPU	PLECO	POPU
L&V [LykourisV'18]	1.34	1.26	1.88	1.78
LMarker [Rohatgi'20]	1.34	1.26	1.88	1.78
LNonMarker [Rohatgi'20]	1.34	1.29	1.88	1.80
RobustFtP	**1.35**	**1.32**	**1.89**	**1.83**
Trust&Doubt	**1.29**	**1.27**	**1.85**	**1.77**

Two *predictors*: simple statistics

Beyond MTS: online matching on the line

Definition by picture: n servers; n online requests



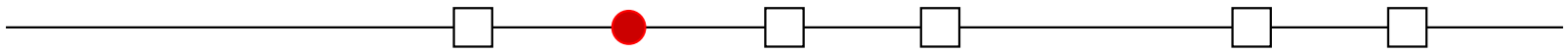
Analoguous prediction setup

- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$*.*

Definition by picture: n servers; n online requests



Analoguous prediction setup

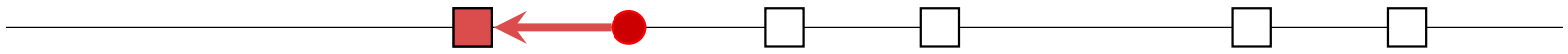
- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$*.*

Beyond MTS: online matching on the line

Definition by picture: n servers; n online requests



Analoguous prediction setup

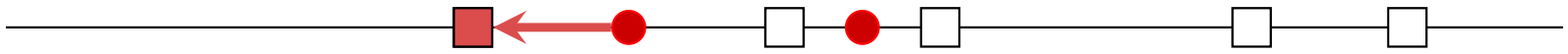
- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$*.*

Beyond MTS: online matching on the line

Definition by picture: n servers; n online requests



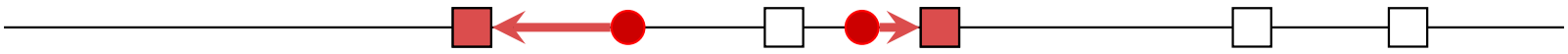
Analoguous prediction setup

- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$*.*

Definition by picture: n servers; n online requests



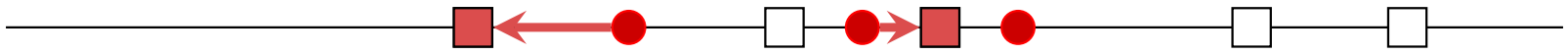
Analoguous prediction setup

- Predict set of servers to be matched so far
- Error: cost of matching between these and Off's matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

RobustFtP*'s analogue is* $O(\min\{1 + \frac{\eta}{\text{Off}}, \log n\})$*-competitive vs* Off.

Definition by picture: n servers; n online requests



Analoguous prediction setup

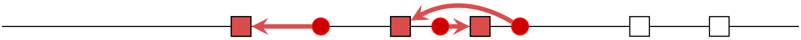
- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$.

Beyond MTS: online matching on the line

Definition by picture: n servers; n online requests



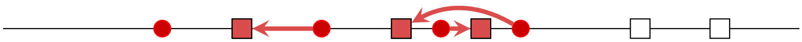
Analoguous prediction setup

- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$.

Definition by picture: n servers; n online requests



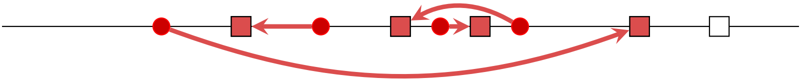
Analoguous prediction setup

- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$*.*

Definition by picture: n servers; n online requests



Analoguous prediction setup

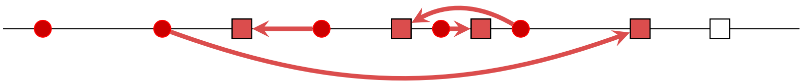
- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$.

Beyond MTS: online matching on the line

Definition by picture: n servers; n online requests



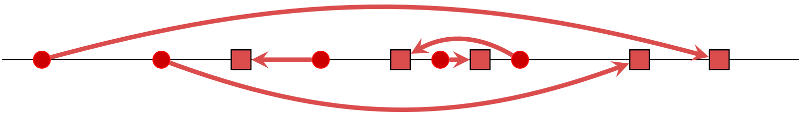
Analoguous prediction setup

- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1 + \frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$.

Definition by picture: n servers; n online requests



Analoguous prediction setup

- Predict set of servers to be matched so far
- Error: cost of matching between these and $\textsc{Off}$'s matched servers

Theorem (Antoniadis, Coester, Elias, Polak, Simon; ICML'20)

$\textsc{RobustFtP}$*'s analogue is* $O(\min\{1+\frac{\eta}{\textsc{Off}}, \log n\})$*-competitive vs* $\textsc{Off}$*.*

Conclusion

MTS (and beyond)

- General prediction setup
- "Optimal" algorithm: RobustFtP

Caching

- Better algorithm: Trust&Doubt
- Less predicted information and better empirical performance than caching-specific setup
- Better than LRU with simple predictor

Perspectives

- Better algorithms for other MTS?
  (e.g., weighted caching, k-server, convex body chasing, graph traversal)