Temporal locality in online algorithms

Maciej Pacut, Mahmoud Parham, Joel Rybicki,
Stefan Schmid, Jukka Suomela, Aleksandr Tereshchenko

DISC
24-28 October 2022

Do online algorithms need access to ALL past requests?

Definition of time-local algorithms

Fix an integer T (a *time horizon*).

An online algorithm is T-time-local if:

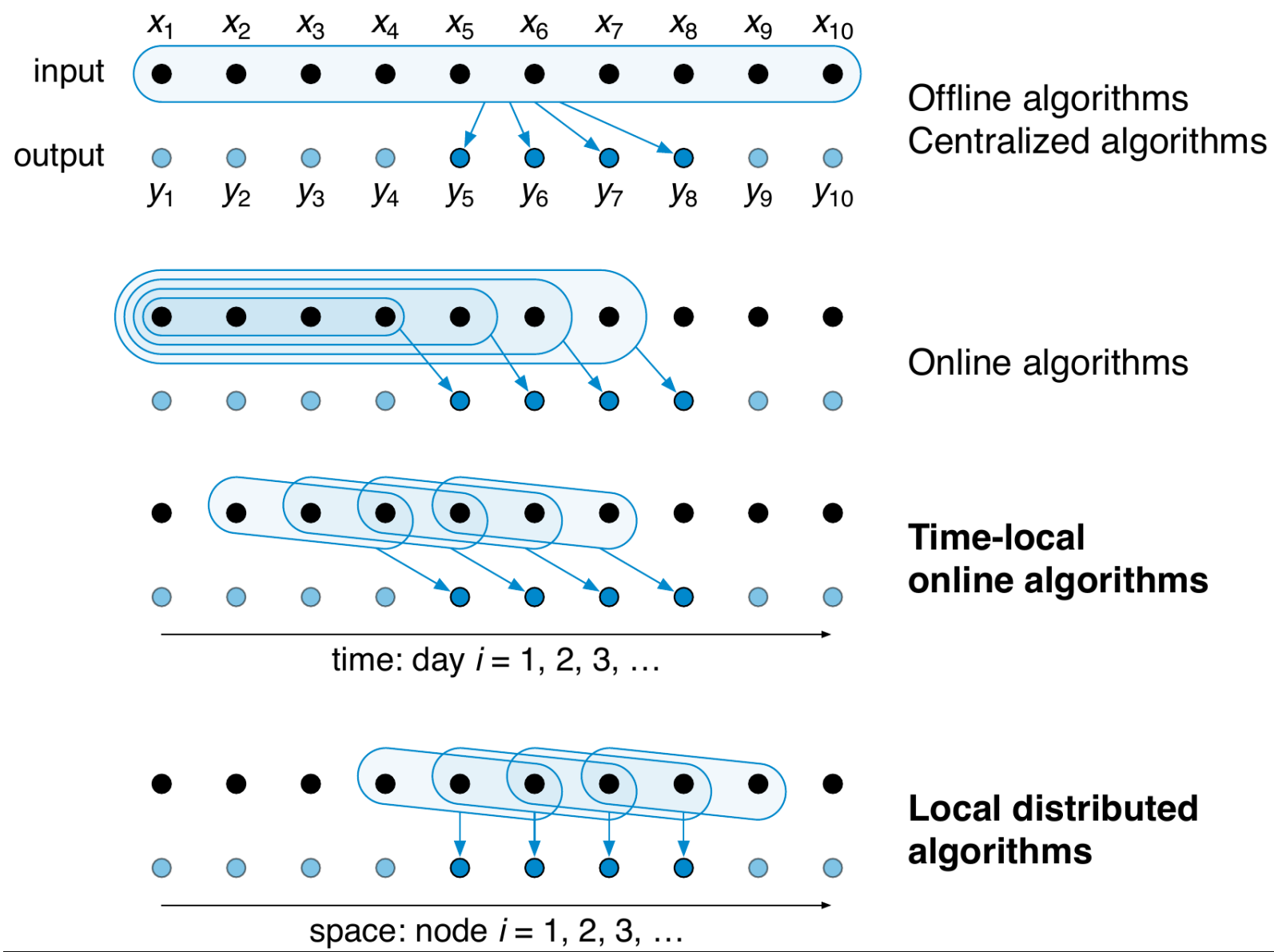
output at time i = function(T last requests)

01 **Online time-local vs distributed algorithms**

02 **Locality in existing online algorithms**

03 **Advantages of time-local**

04 **How competitive?**


x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}
input
output
y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10}
Offline algorithms
Centralized algorithms
Online algorithms
Time-local
online algorithms
time: day i = 1, 2, 3, …
Local distributed
algorithms
space: node i = 1, 2, 3, …

Are existing online algorithms time-local?

NO: Some algorithms strongly rely on all past
(e.g. work function algorithms for k-server, metrical task system)

YES: Some phase-based algorithms
(e.g. Move-to-Min for file migration)

YES: For some problems, configuration can be reset periodically
(e.g. algorithms for list access, binary search trees)

Why time-local online algorithms?

Synthesis

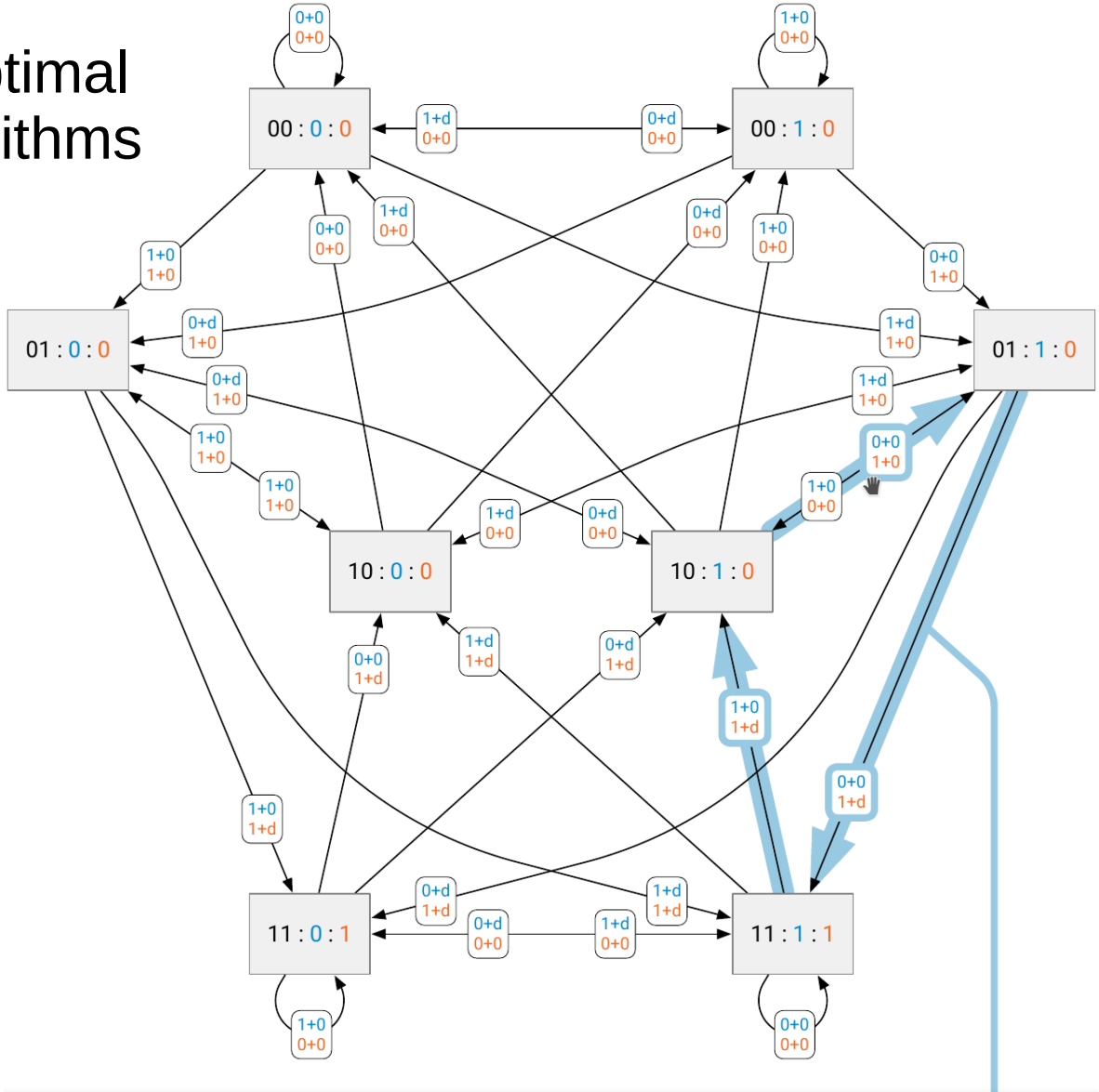
Automated synthesis of optimal time-local algorithms

Fault-recovery

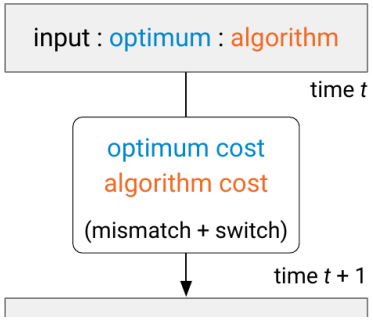
Consistent decisions in T rounds after a crash

Simplicity

Synthesis of optimal time-local algorithms



Notation:



How competitive time-local algorithms are?

1. File migration: constant-competitive

2. Caching: non-competitive

3. Translation theorem for *bounded-monotone* problems
   (e.g. list access, binary search trees)

full-history online c-competitive algorithm
translated to
time-local c*(1+epsilon)-competitive

Contributions

- characterizing locality in online algorithms
- synthesis
- full-history algorithms to time-local algorithms
- tight algorithms for file migration and caching

Full paper: https://arxiv.org/abs/2102.09413