

# An Exact Algorithm for the Linear Tape Scheduling Problem

Valentin Honoré, **Bertrand Simon**, Frédéric Suter

IN2P3 Computing Center / CNRS, Villeurbanne

Aussois – May 2022



AND NOW,  
MAGNETIC TAPES!



# Tape usage today



(or 11  football fields)

≈ 20TB on 1000s × 1km read at 10m/s – 100s MB/s

<https://commons.wikimedia.org/wiki/File:LT02-cart-wo-top-shell.jpg>  
[https://commons.wikimedia.org/wiki/File:Usain\\_Bolt\\_Rio\\_100m\\_final\\_2016i-cr.jpg](https://commons.wikimedia.org/wiki/File:Usain_Bolt_Rio_100m_final_2016i-cr.jpg)



Primordial for HTC (High Throughput Computing)

e.g.,

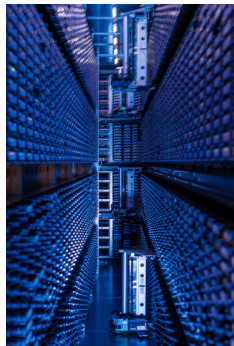


(100s PB)

also: media companies, cloud archive. . .

😊 Impressive technology improvements  
 density: + 30% / year (vs HDD: + 8%)

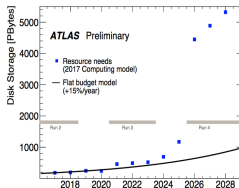
😞 high latency (mount, load, position → few mn)  
 Adapted for Write Once Read Many



# Why not use hard drives?



up to 6-10 times cheaper overall (before 2020)



[Xin Zhao, HEPIX 2018]

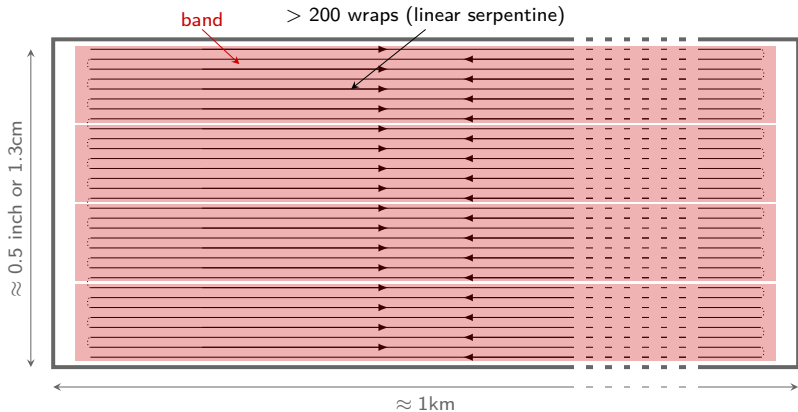


air gap, power failure, lifetime



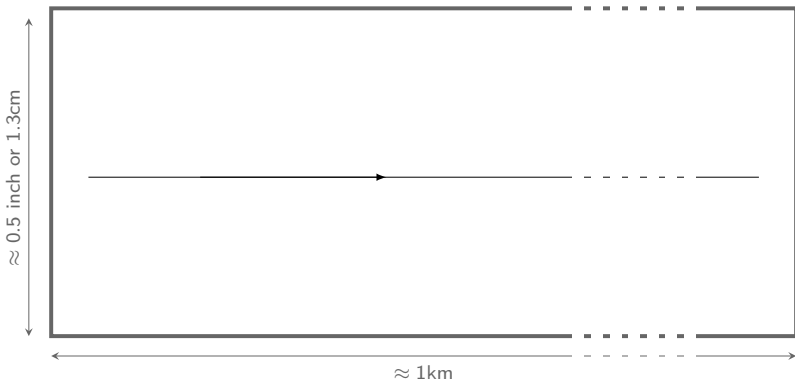
energy-efficient

# Overview of a tape



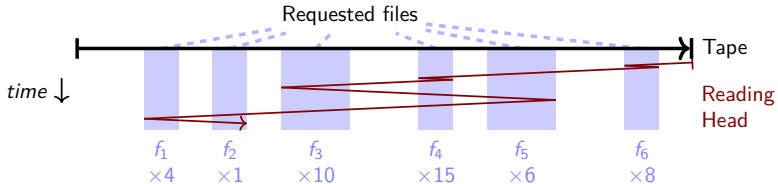
*wrap = dozens of tracks read / written simultaneously by parallel heads*

# Overview of our tape model



# Linear Tape Scheduling Problem

[CardonhaReal'16]



## Assumptions:

- ▶ files are read left-to-right
- ▶ start on the right
- ▶ constant speed

## Input:

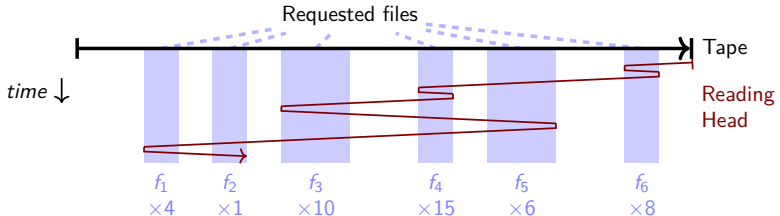
- ▶ tape of  $n_f$  consecutive files
- ▶  $n$  file requests (44 here)
- ▶  $n_{req}$  distinct files requested (6)

Objective: average service time

Motivation: lack of fundamental theoretical results, models local files

# Linear Tape Scheduling Problem

[CardonhaReal'16]



## Assumptions:

- ▶ files are read left-to-right
- ▶ start on the right
- ▶ constant speed
- ▶ [new] U-turn penalty  $U$

## Input:

- ▶ tape of  $n_f$  consecutive files
- ▶  $n$  file requests (44 here)
- ▶  $n_{req}$  distinct files requested (6)

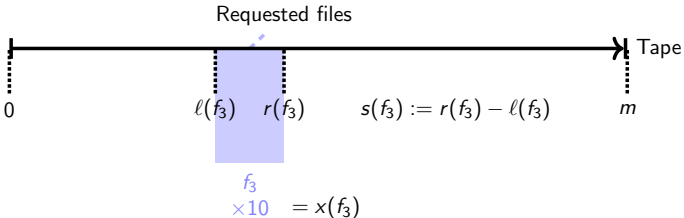
Objective: average service time

Motivation: lack of fundamental theoretical results, models local files



# Linear Tape Scheduling Problem

[CardonhaReal'16]



## Assumptions:

- ▶ files are read left-to-right
- ▶ start on the right
- ▶ constant speed
- ▶ [new] U-turn penalty  $U$

## Input:

- ▶ tape of  $n_f$  consecutive files
- ▶  $n$  file requests (44 here)
- ▶  $n_{req}$  distinct files requested (6)

Objective: average service time

Motivation: lack of fundamental theoretical results, models local files

# Related to the Linear Tape Scheduling Problem



## Travelling Salesperson Problem (TSP)

- ▶ super-famous NP-hard problem
- ▶ recent  $(1.5 - 10^{-36})$ -approximation [KarlínKG'21]
- ▶ 😞 minimizes makespan, trivial on the real line

## Minimum Latency Problem / TRP (Repair) - variant

- ▶ 😊 minimize average service time  $\in P$  on the real line
- ▶ delays to repair a node: complexity open



## Dial-a-ride variant on the real line

- ▶  $\approx$  LTSP but with overlapping files in both directions  
→ NP-hard

Tapes except LTSP: 2 specific experimental papers in the 90's

# Structural results

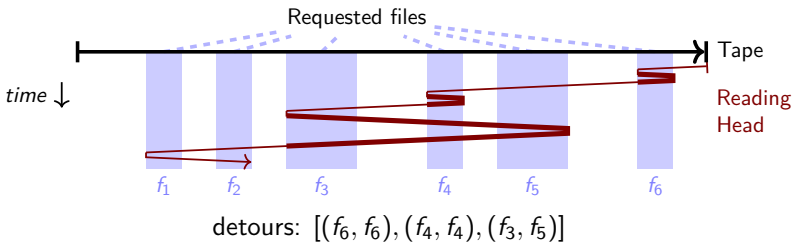
## Any optimal solution

- ▶ after reaching  $\ell(f_1)$ , go straight to the rightmost unread request
- ▶ can be described by a set of **detours** done before

### Definition (Detours)

A solution includes the **detour**  $(a,b)$  with  $a \leq b$  if:

- ▶ the 1st time the head reaches  $\ell(a)$ , go straight to  $r(b)$ , back to  $\ell(a)$



# Structural results

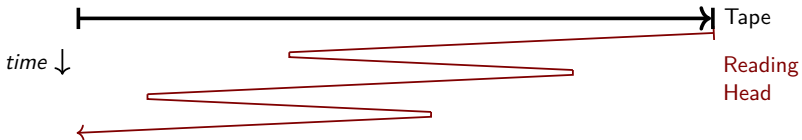
## Any optimal solution

- ▶ after reaching  $\ell(f_1)$ , go straight to the rightmost unread request
- ▶ can be described by a set of **detours** done before

### Definition (Detours)

A solution includes the **detour**  $(a,b)$  with  $a \leq b$  if:

- ▶ the 1st time the head reaches  $\ell(a)$ , go straight to  $r(b)$ , back to  $\ell(a)$



**Lemma:** detours never partially overlap (*strictly laminar*)

# Structural results

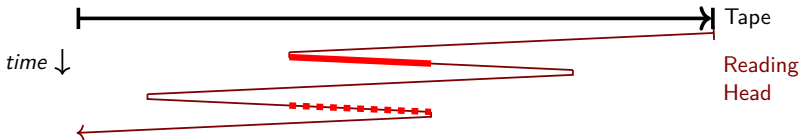
## Any optimal solution

- ▶ after reaching  $\ell(f_1)$ , go straight to the rightmost unread request
- ▶ can be described by a set of **detours** done before

### Definition (Detours)

A solution includes the **detour**  $(a,b)$  with  $a \leq b$  if:

- ▶ the 1st time the head reaches  $\ell(a)$ , go straight to  $r(b)$ , back to  $\ell(a)$

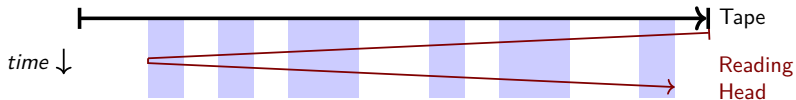


**Lemma:** detours never partially overlap (*strictly laminar*)

# Naive algorithms

**NO DETOUR:** go to the leftmost request, then to the rightmost

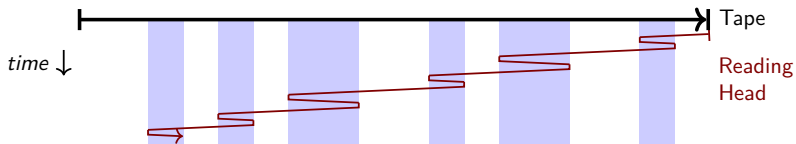
can be arbitrarily bad (place urgent requests on the right)



**GS (Greedy Schedule):** do all atomic detours, *i.e.*,  $\{(f_i, f_i)\}_{\forall i}$

Lemma [CardonhaReal'16]: **GS** is a 3-approximation if  $U = 0$

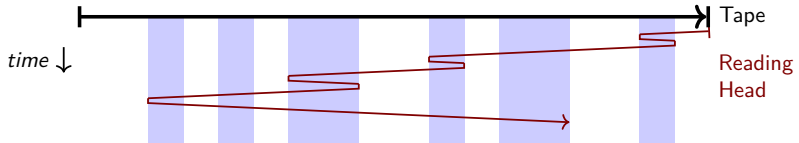
Proof: does  $\leq 3$  times the optimal distance before reading each request



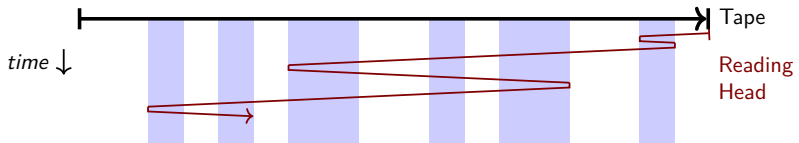
# Heuristic improvements

# [CardonhaCiréReal'18]

**FGS (Filtered)**: remove detrimental atomic detours in  $O(n_{req}^2)$



**NFGS (Non-atomic)**: greedily add long detours if currently beneficial. Make one pass from left to right. Complexity in  $O(n_{req}^3)$



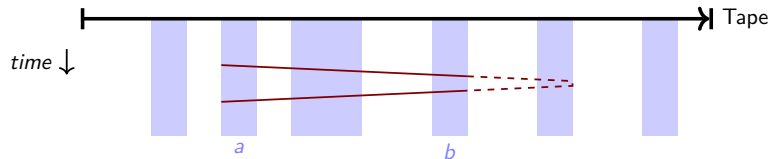
# Dynamic Program: overview

**Each cell: three parameters**  $T[a, b, n_{skip}]$

► compute the best strategy from  $r(\mathbf{b})$  to  $\ell(\mathbf{a})$  assuming:

- 1 there is a detour  $(\mathbf{a}, f)$  for some  $f \geq \mathbf{b}$ ,
- 2 there is no detour  $(f_1, f_2)$  such that  $\mathbf{a} < f_1 < \mathbf{b} < f_2$ ,
- 3 when reaching  $r(\mathbf{b})$ , exactly  $n_{skip}$  requests have been skipped.

⇒ value  $\approx$  cost contribution from 'first  $r(b)$ ' to ' $r(b)$  after reading  $a$ '



Subtleties:  $\forall$  request on  $f$ , do not count the cost  $m \rightarrow \ell(f) \rightarrow r(f)$   
if  $f$  is read after  $b$ , remove one  $U$  (counted before)



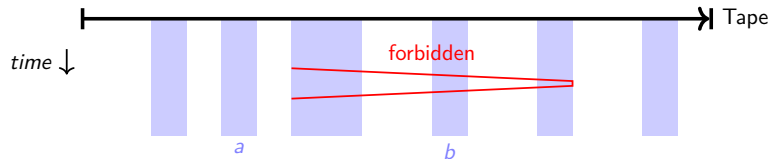
# Dynamic Program: overview

**Each cell: three parameters**  $T[a, b, n_{skip}]$

► compute the best strategy from  $r(\mathbf{b})$  to  $\ell(\mathbf{a})$  assuming:

- 1 there is a detour  $(\mathbf{a}, f)$  for some  $f \geq \mathbf{b}$ ,
- 2 there is no detour  $(f_1, f_2)$  such that  $\mathbf{a} < f_1 < \mathbf{b} < f_2$ ,
- 3 when reaching  $r(\mathbf{b})$ , exactly  $n_{skip}$  requests have been skipped.

⇒ value  $\approx$  cost contribution from 'first  $r(b)$ ' to ' $r(b)$  after reading  $a$ '



Subtleties:  $\forall$  request on  $f$ , do not count the cost  $m \rightarrow \ell(f) \rightarrow r(f)$   
if  $f$  is read after  $b$ , remove one  $U$  (counted before)

# Dynamic Program: overview

**Each cell: three parameters**  $T[a, b, n_{skip}]$

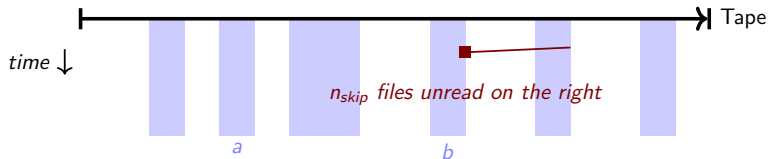
► compute the best strategy from  $r(\mathbf{b})$  to  $\ell(\mathbf{a})$  assuming:

1 there is a detour  $(\mathbf{a}, f)$  for some  $f \geq \mathbf{b}$ ,

2 there is no detour  $(f_1, f_2)$  such that  $\mathbf{a} < f_1 < \mathbf{b} < f_2$ ,

3 when reaching  $r(\mathbf{b})$ , exactly  $n_{skip}$  requests have been skipped.

⇒ value  $\approx$  cost contribution from 'first  $r(\mathbf{b})$ ' to ' $r(\mathbf{b})$  after reading  $\mathbf{a}$ '



Subtleties:  $\forall$  request on  $f$ , do not count the cost  $m \rightarrow \ell(f) \rightarrow r(f)$   
if  $f$  is read after  $b$ , remove one  $U$  (counted before)

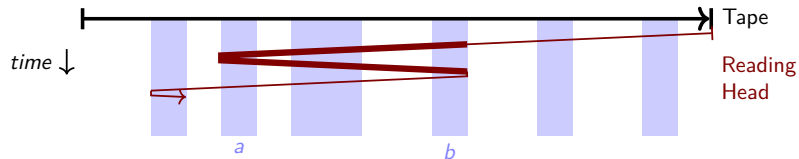
# Dynamic Program: overview

**Each cell: three parameters**  $T[a, b, n_{skip}]$

► compute the best strategy from  $r(\mathbf{b})$  to  $\ell(\mathbf{a})$  assuming:

- 1 there is a detour  $(\mathbf{a}, f)$  for some  $f \geq \mathbf{b}$ ,
- 2 there is no detour  $(f_1, f_2)$  such that  $\mathbf{a} < f_1 < \mathbf{b} < f_2$ ,
- 3 when reaching  $r(\mathbf{b})$ , exactly  $n_{skip}$  requests have been skipped.

⇒ value  $\approx$  cost contribution from 'first  $r(b)$ ' to ' $r(b)$  after reading  $a$ '



Subtleties:  $\forall$  request on  $f$ , do not count the cost  $m \rightarrow \ell(f) \rightarrow r(f)$   
if  $f$  is read after  $b$ , remove one  $U$  (counted before)

# Dynamic Program: overview

**Each cell: three parameters**  $T[a, b, n_{skip}]$

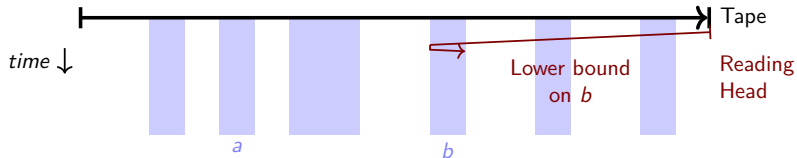
► compute the best strategy from  $r(\mathbf{b})$  to  $\ell(\mathbf{a})$  assuming:

1 there is a detour  $(\mathbf{a}, f)$  for some  $f \geq \mathbf{b}$ ,

2 there is no detour  $(f_1, f_2)$  such that  $\mathbf{a} < f_1 < \mathbf{b} < f_2$ ,

3 when reaching  $r(\mathbf{b})$ , exactly  $n_{skip}$  requests have been skipped.

⇒ value  $\approx$  cost contribution from 'first  $r(b)$ ' to ' $r(b)$  after reading  $a$ '



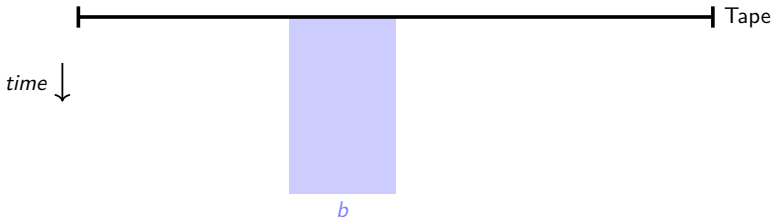
Subtleties:  $\forall$  request on  $f$ , do not count the cost  $m \rightarrow \ell(f) \rightarrow r(f)$   
if  $f$  is read after  $b$ , remove one  $U$  (counted before)

# Dynamic program: base case, $a = b$

$a = b \implies$  **a detour starts at  $\ell(b)$**

$n_\ell(b) :=$  # file requests strictly on the left of  $b$

$$T[b, b, n_{skip}] = 2 \cdot s(b) \cdot (n_{skip} + n_\ell(b))$$

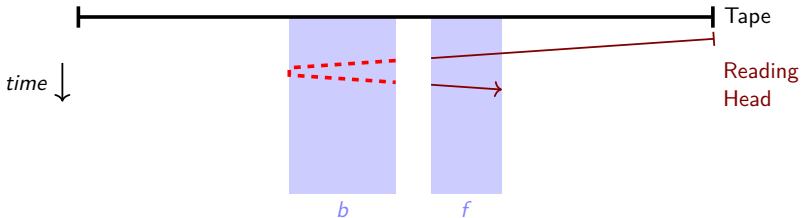


# Dynamic program: base case, $a = b$

$a = b \implies$  **a detour starts at  $\ell(b)$**

$n_\ell(b) := \#$  file requests strictly on the left of  $b$

$$T[b, b, n_{skip}] = 2 \cdot s(b) \cdot (n_{skip} + n_\ell(b))$$

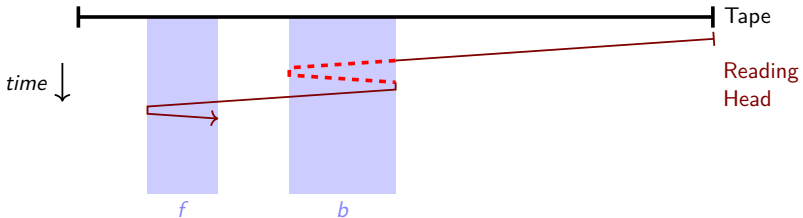


# Dynamic program: base case, $a = b$

$a = b \implies$  **a detour starts at  $\ell(b)$**

$n_\ell(b) := \#$  file requests strictly on the left of  $b$

$$T[b, b, n_{skip}] = 2 \cdot s(b) \cdot (n_{skip} + n_\ell(b))$$

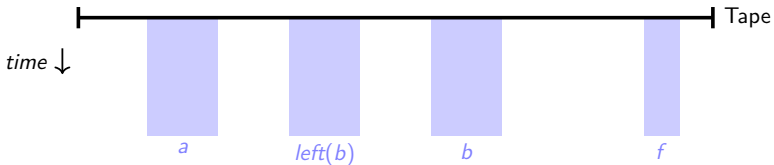


# Dynamic program: inductive case 1, skip $b$

$a < b$  and assume  $b$  is skipped

$left(f) :=$  requested file directly on the left of  $f$

$$\begin{aligned} skip(a, b, n_{skip}) := & T[a, left(b), n_{skip} + x(b)] \\ & + 2 \cdot (r(b) - r(left(b))) \cdot (n_{skip} + n_\ell(a)) \\ & + 2 \cdot (\ell(b) - r(left(b))) \cdot x(b) \end{aligned}$$



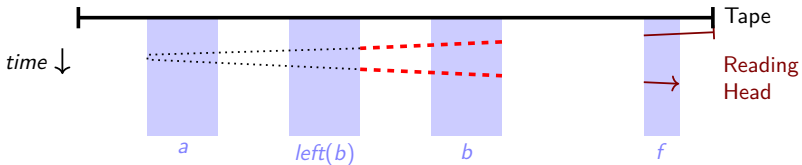


# Dynamic program: inductive case 1, skip $b$

$a < b$  and assume  $b$  is skipped

$left(f) :=$  requested file directly on the left of  $f$

$$\begin{aligned}
 skip(a, b, n_{skip}) := & T[a, left(b), n_{skip} + x(b)] \\
 & + 2 \cdot (r(b) - r(left(b))) \cdot (n_{skip} + n_\ell(a)) \\
 & + 2 \cdot (\ell(b) - r(left(b))) \cdot x(b)
 \end{aligned}$$

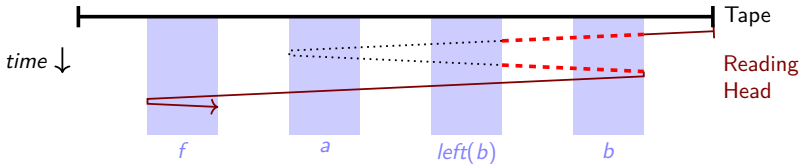


# Dynamic program: inductive case 1, skip $b$

$a < b$  and assume  $b$  is skipped

$left(f) :=$  requested file directly on the left of  $f$

$$\begin{aligned}
 skip(a, b, n_{skip}) := & T[a, left(b), n_{skip} + x(b)] \\
 & + 2 \cdot (r(b) - r(left(b))) \cdot (n_{skip} + n_\ell(a)) \\
 & + 2 \cdot (\ell(b) - r(left(b))) \cdot x(b)
 \end{aligned}$$

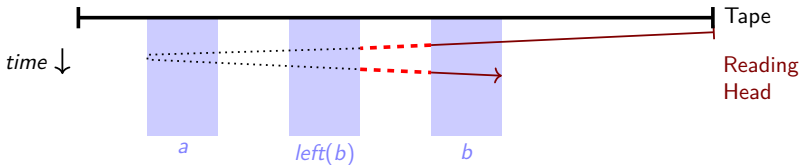


# Dynamic program: inductive case 1, skip $b$

$a < b$  and assume  $b$  is skipped

$left(f) :=$  requested file directly on the left of  $f$

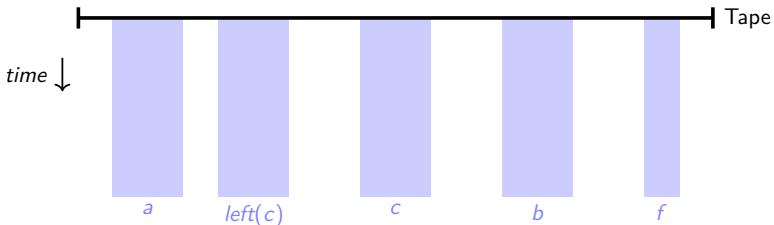
$$\begin{aligned}
 skip(a, b, n_{skip}) := & T[a, left(b), n_{skip} + x(b)] \\
 & + 2 \cdot (r(b) - r(left(b))) \cdot (n_{skip} + n_{\ell}(a)) \\
 & + 2 \cdot (\ell(b) - r(left(b))) \cdot x(b)
 \end{aligned}$$



# Dynamic program: inductive case 2, intertwined detours

$a < b$ , there is a detour  $(c, b)$  with  $c > a$

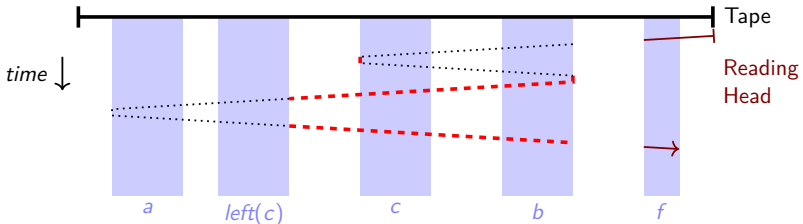
$$\begin{aligned} \text{detour}_c(a, b, n_{\text{skip}}) &:= T[a, \text{left}(c), n_{\text{skip}}] + T[c, b, n_{\text{skip}}] \\ &\quad + 2 \cdot (r(b) - r(\text{left}(c))) \cdot (n_{\text{skip}} + n_\ell(a)) \\ &\quad + 2 \cdot U \cdot (n_{\text{skip}} + n_\ell(c)) \end{aligned}$$



# Dynamic program: inductive case 2, intertwined detours

$a < b$ , there is a detour  $(c, b)$  with  $c > a$

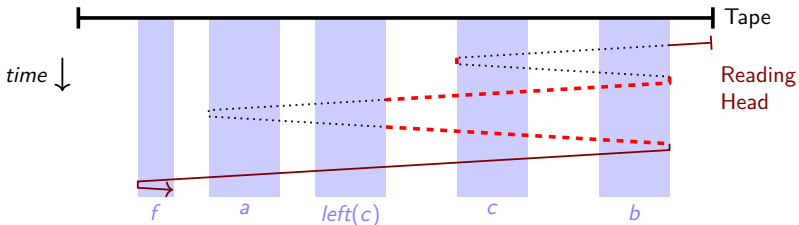
$$\begin{aligned} \text{detour}_c(a, b, n_{\text{skip}}) &:= T[a, \text{left}(c), n_{\text{skip}}] + T[c, b, n_{\text{skip}}] \\ &\quad + 2 \cdot (r(b) - r(\text{left}(c))) \cdot (n_{\text{skip}} + n_\ell(a)) \\ &\quad + 2 \cdot U \cdot (n_{\text{skip}} + n_\ell(c)) \end{aligned}$$



# Dynamic program: inductive case 2, intertwined detours

$a < b$ , there is a detour  $(c, b)$  with  $c > a$

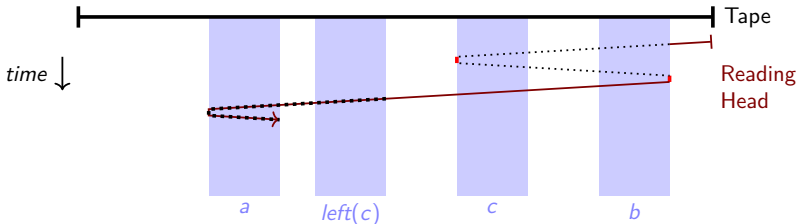
$$\begin{aligned} \text{detour}_c(a, b, n_{\text{skip}}) &:= T[a, \text{left}(c), n_{\text{skip}}] + T[c, b, n_{\text{skip}}] \\ &\quad + 2 \cdot (r(b) - r(\text{left}(c))) \cdot (n_{\text{skip}} + n_\ell(a)) \\ &\quad + 2 \cdot U \cdot (n_{\text{skip}} + n_\ell(c)) \end{aligned}$$



# Dynamic program: inductive case 2, intertwined detours

$a < b$ , there is a detour  $(c, b)$  with  $c > a$

$$\begin{aligned} \text{detour}_c(a, b, n_{\text{skip}}) &:= T[a, \text{left}(c), n_{\text{skip}}] + T[c, b, n_{\text{skip}}] \\ &\quad + 2 \cdot (r(b) - r(\text{left}(c))) \cdot (n_{\text{skip}} + n_\ell(a)) \\ &\quad + 2 \cdot U \cdot (n_{\text{skip}} + n_\ell(c)) \end{aligned}$$



# Dynamic program: complete formulation

$$\begin{aligned} \text{skip}(a, b, n_{\text{skip}}) &:= T[a, \text{left}(b), n_{\text{skip}} + x(b)] \\ &\quad + 2 \cdot (r(b) - r(\text{left}(b))) \cdot (n_{\text{skip}} + n_{\ell}(a)) \\ &\quad + 2 \cdot (\ell(b) - r(\text{left}(b))) \cdot x(b) \end{aligned}$$

$$\begin{aligned} \text{detour}_c(a, b, n_{\text{skip}}) &:= T[a, \text{left}(c), n_{\text{skip}}] + T[c, b, n_{\text{skip}}] \\ &\quad + 2 \cdot (r(b) - r(\text{left}(c))) \cdot (n_{\text{skip}} + n_{\ell}(a)) \\ &\quad + 2 \cdot U \cdot (n_{\text{skip}} + n_{\ell}(c)) \end{aligned}$$

define  $F_{a,b} :=$  files requested between  $a$  and  $b$  excluding  $a$

Dynamic program DP (with  $a < b$ )

$$T[b, b, n_{\text{skip}}] = 2 \cdot s(b) \cdot (n_{\text{skip}} + n_{\ell}(b))$$

$$T[a, b, n_{\text{skip}}] = \min \left( \text{skip}(a, b, n_{\text{skip}}) ; \min_{c \in F_{a,b}} \text{detour}_c(a, b, n_{\text{skip}}) \right)$$

Theorem

**DP** solves LTSP exactly in time  $O(n \cdot n_{\text{req}}^3)$ .



# More dynamic programs

**LOGDP( $\lambda$ ):** DP restricted to detours spanning  $\lambda \log n_{req}$  requested files

*Reduced complexity in  $O(\lambda^2 \cdot n_{req} \cdot n \cdot \log^2(n_{req}))$ , tested with  $\lambda \in \{1, 5\}$*

**SIMPLEDP:** DP forbidding intertwined (*i.e.*, overlapping) detours

Similar to DP but  $a$  is always  $f_1$ : no need to call  $T[b, c, n_{skip}]$

*Reduced complexity in  $O(n \cdot n_{req}^2)$*

**Lemma:** for all  $U$ , competitive ratio  $\in [\frac{5}{3}, 3]$

Note: dependency in  $n$  and not  $\log n \rightarrow$  pseudo-polynomial for high-multiplicity instances (harder problem as in scheduling)

Note2: concurrent similar solution from [CardonhaCireReal'21]

# Simulations: overview

## Dataset: 2 weeks at CC-IN2P3

- ▶ 169 tapes, > 3M files
- ▶ focus on reading operations
- ▶ filtering steps, data processing (e.g., merge reads on aggregates)
- ▶ median data: 150 files requested, 3k requests, 50% file size variation

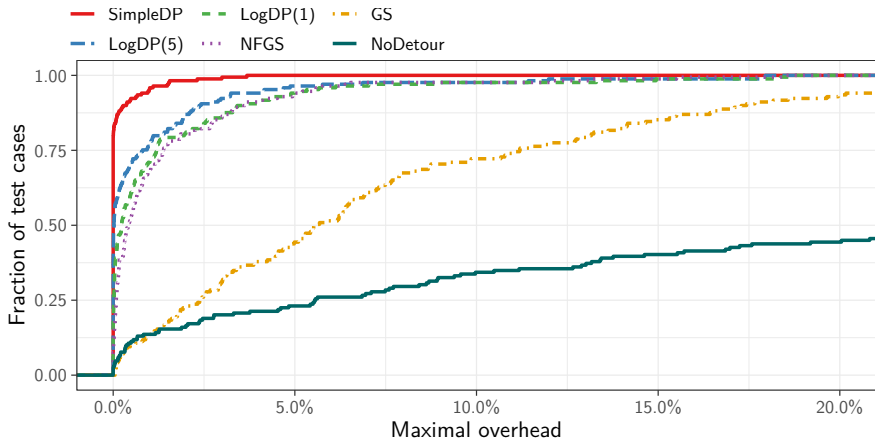
Code + dataset (with statistical descriptions) available online

## Experimental methodology

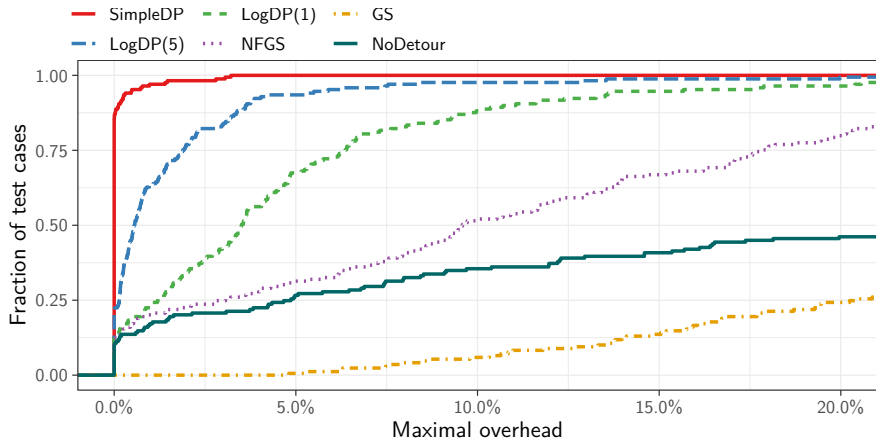
- ▶ choose 3 values for  $U$ :  $\{0, 0.5, 1\} \times$  average file size reading time
- ▶ median time performance (seconds, on a simple Python program):

FGS	NFGS	LOGDP(1)	SIMPLEDP	LOGDP(5)	DP
< 0.1	0.5	5	20	50	280

# Simulation results, $U = 0$



Performance profile: best is top-left (most instances with low overhead vs  $OPT$ )

Simulation results,  $U = \text{file}$ 

Performance profile: best is top-left (most instances with low overhead vs  $OPT$ )

# Conclusion



<https://commons.wikimedia.org/wiki/File:LT02-cart-wo-top-shell.jpg>

## General: tapes are past & future

- ▶ tapes stay primordial in some fields (€\$£¥₩) but neglected by CS research
- ▶ fundamental problems are still open

## On LTSP

- ▶ high-multiplicity variant remains open
- ▶ huge gap between theoretically studied models and practical heuristics

## Perspectives on other tape-related topics

- ▶ multi-tape requests: optimize waiting queues
- ▶ optimize tape / disk storage ratio

