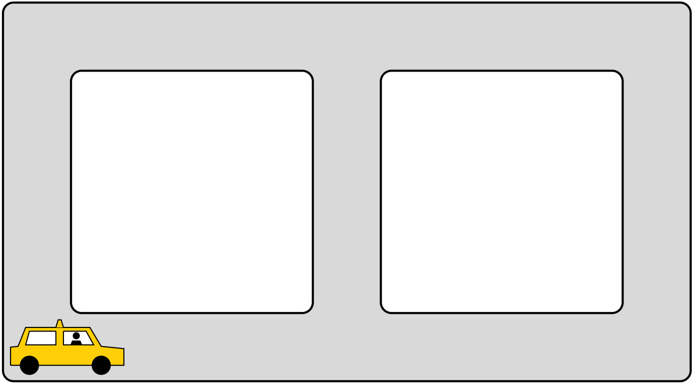
Tight Analysis of the Lazy Algorithm for Open Online Dial-a-Ride

Júlia Baligács, Yann Disser, Farehe Soheil, David Weckbecker

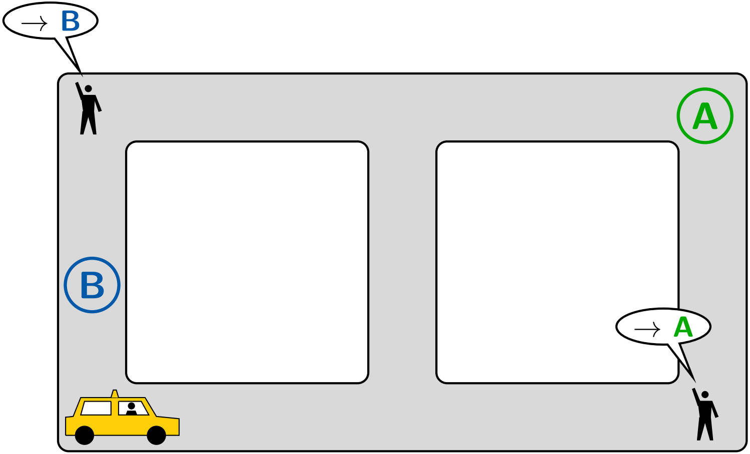
TU Darmstadt, Germany

We acknowledge funding by DFG through grant DI 2041/2.

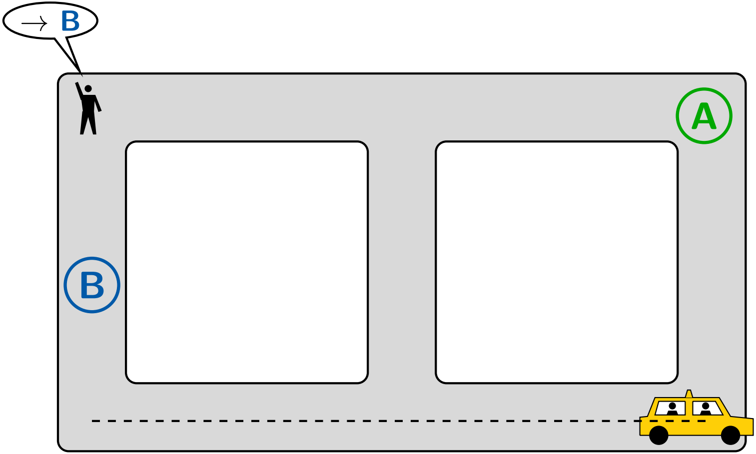
Example: taxi driver



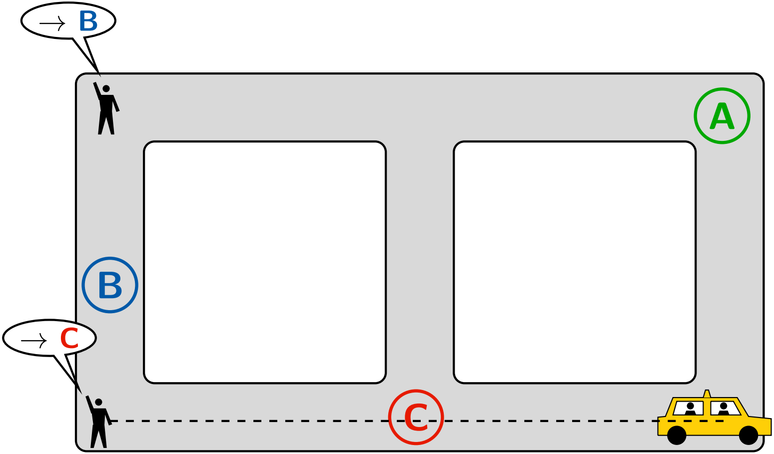
Example: taxi driver



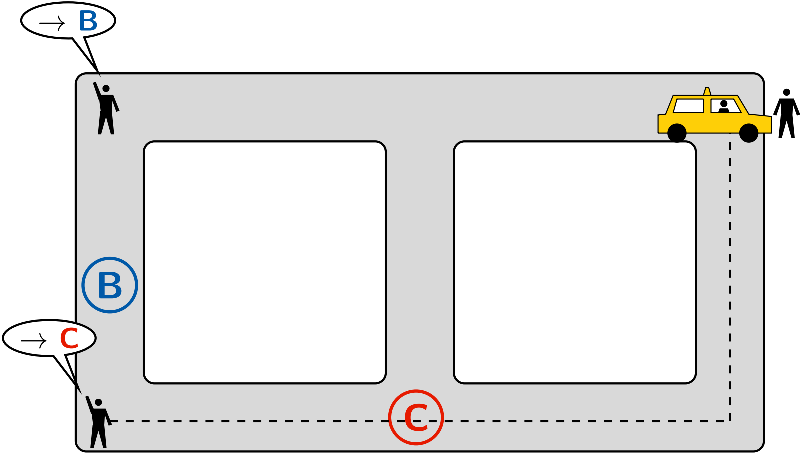
Example: taxi driver



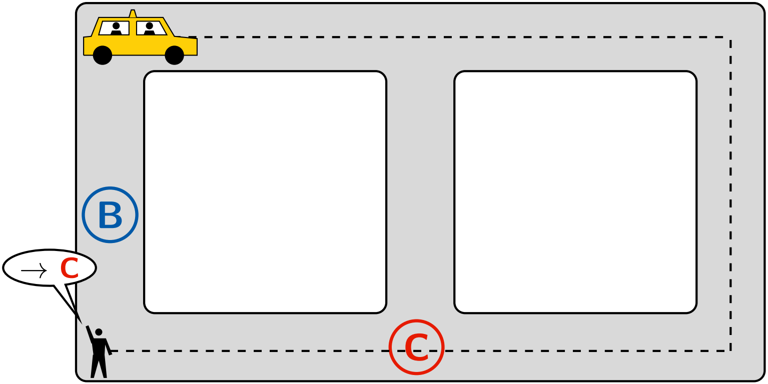
Example: taxi driver



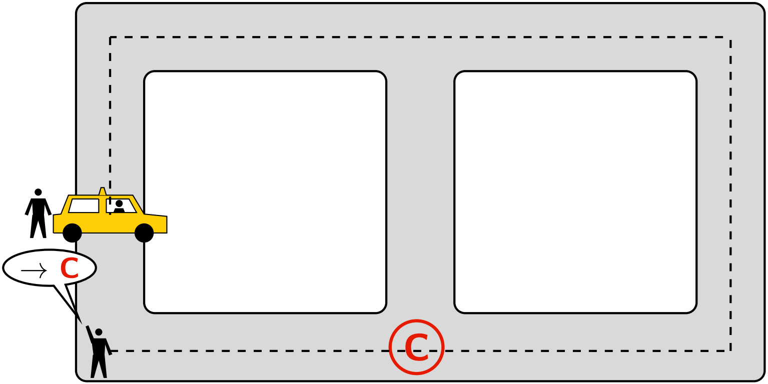
Example: taxi driver



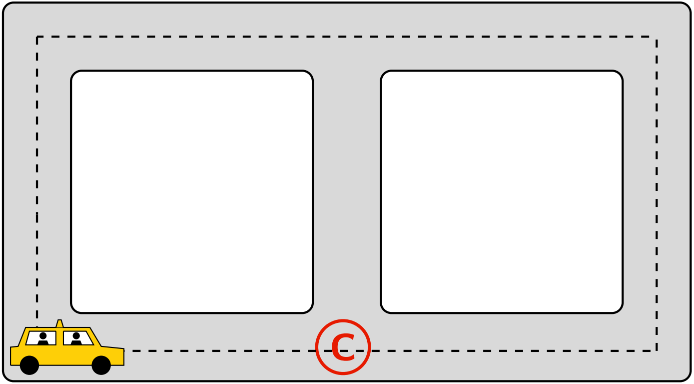
Example: taxi driver



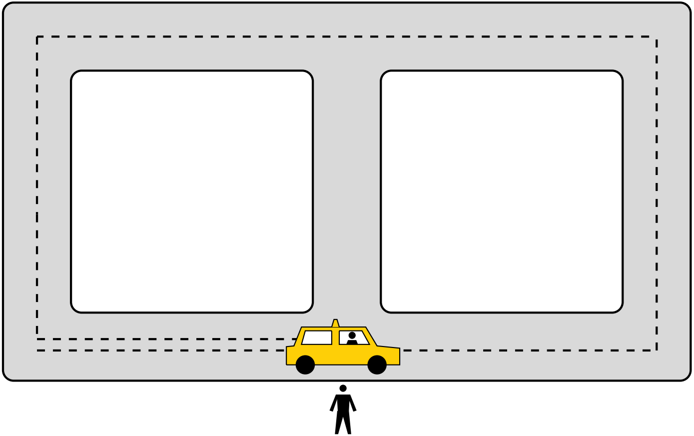
Example: taxi driver



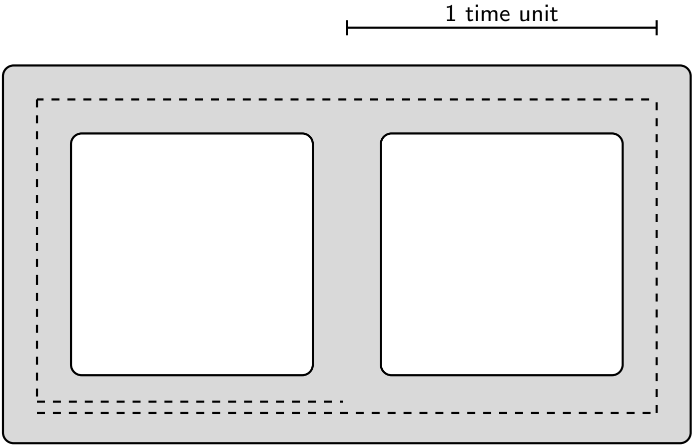
Example: taxi driver



Example: taxi driver

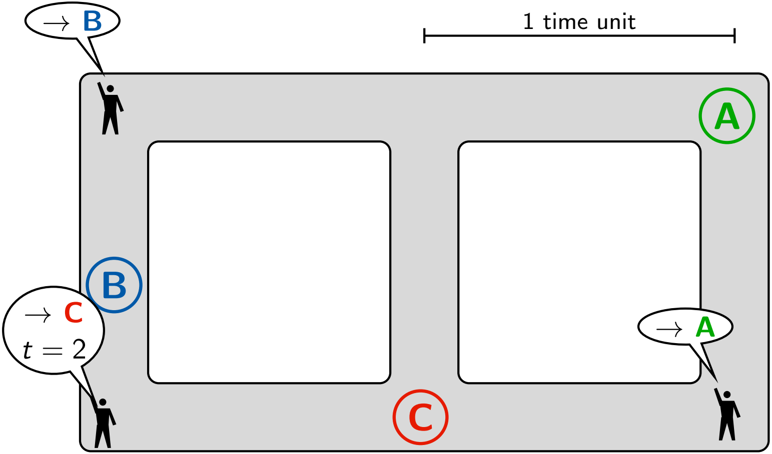


Example: taxi driver



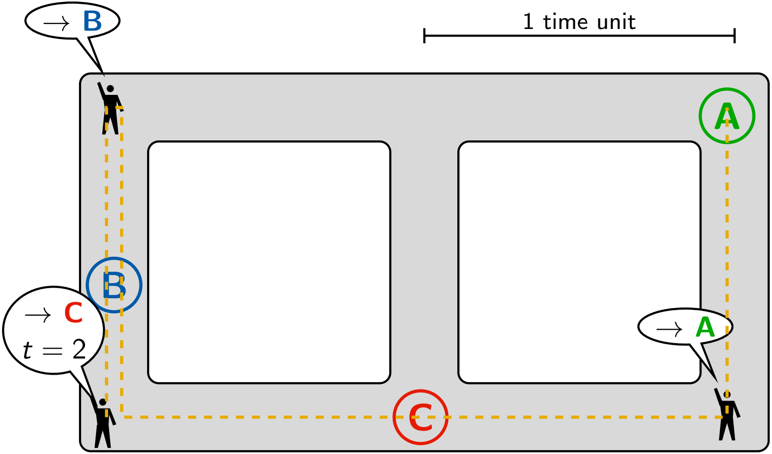
- our solution: completed after 7 time units

Example: taxi driver



- our solution: completed after 7 time units

Example: taxi driver

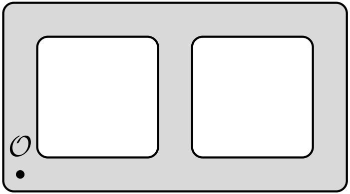


- our solution: completed after 7 time units
- optimal solution: completed after 5 time units

The online dial-a-ride problem

setting:

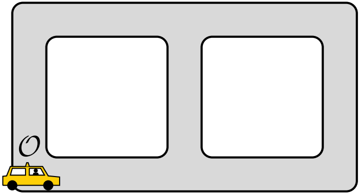
- metric space (M, d) with origin $\mathcal{O}$



The online dial-a-ride problem

setting:

- metric space (M, d) with origin $\mathcal{O}$
- single server with capacity $c \in \mathbb{N}$
  - can move with speed ≤ 1
  - initially located at $\mathcal{O}$



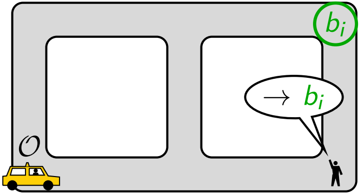
The online dial-a-ride problem

setting:

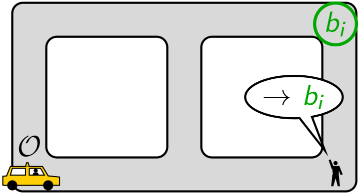
- metric space (M, d) with origin $\mathcal{O}$
- single server with capacity $c \in \mathbb{N}$
  - can move with speed ≤ 1
  - initially located at $\mathcal{O}$

over time:

- requests $r_i = (a_i, b_i; t_i)$ are revealed
  - pick up at a_i after time t_i
  - deliver at b_i



The online dial-a-ride problem



setting:

- metric space (M, d) with origin $\mathcal{O}$
- single server with capacity $c \in \mathbb{N}$
  - can move with speed ≤ 1
  - initially located at $\mathcal{O}$

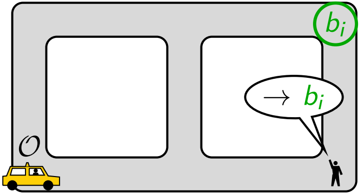
over time:

- requests $r_i = (a_i, b_i; t_i)$ are revealed
  - pick up at a_i after time t_i
  - deliver at b_i

objective:

- minimize completion time (without returning to origin)

The online dial-a-ride problem



setting:

- metric space (M, d) with origin $\mathcal{O}$
- single server with capacity $c \in \mathbb{N}$
  - can move with speed ≤ 1
  - initially located at $\mathcal{O}$

over time:

- requests $r_i = (a_i, b_i; t_i)$ are revealed
  - pick up at a_i after time t_i
  - deliver at b_i

objective:

- minimize completion time (without returning to origin)

online algorithm:
learns r_i at time t_i

offline optimum:
knows all requests in advance

Competitive analysis

Definition: for an online algorithm $\textsc{Alg}$

a) $\textsc{Alg}(\sigma)$: completion time on request sequence σ

Competitive analysis

Definition: for an online algorithm $\textsc{Alg}$

a) $\textsc{Alg}(\sigma)$: completion time on request sequence σ

b) $\textsc{Opt}(\sigma)$: completion time of offline optimum on σ

Competitive analysis

Definition: for an online algorithm $\textsc{Alg}$

a) $\textsc{Alg}(\sigma)$: completion time on request sequence σ

b) $\textsc{Opt}(\sigma)$: completion time of offline optimum on σ

c) $\textsc{Alg}$ is ρ-**competitive** if $\textsc{Alg}(\sigma) \leq \rho \cdot \textsc{Opt}(\sigma)$ for all σ.

Competitive analysis

Definition: for an online algorithm $\textsc{Alg}$

a) $\textsc{Alg}(\sigma)$: completion time on request sequence σ

b) $\textsc{Opt}(\sigma)$: completion time of offline optimum on σ

c) $\textsc{Alg}$ is ρ-**competitive** if $\textsc{Alg}(\sigma) \leq \rho \cdot \textsc{Opt}(\sigma)$ for all σ.

d) The **competitive ratio** of $\textsc{Alg}$ is $\inf\{\rho : \textsc{Alg} \text{ is } \rho\text{-competitive}\}$.

Competitive analysis

Definition: for an online algorithm $\textsc{Alg}$

a) $\textsc{Alg}(\sigma)$: completion time on request sequence σ

b) $\textsc{Opt}(\sigma)$: completion time of offline optimum on σ

c) $\textsc{Alg}$ is ρ-**competitive** if $\textsc{Alg}(\sigma) \leq \rho \cdot \textsc{Opt}(\sigma)$ for all σ.

d) The **competitive ratio** of $\textsc{Alg}$ is $\inf\{\rho : \textsc{Alg} \text{ is } \rho\text{-competitive}\}$.

Question: What is the best possible competitive ratio for the online dial-a-ride problem?

Some simple algorithms

IGNORE:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

Some simple algorithms

IGNORE:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

Example:

Some simple algorithms

IGNORE:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

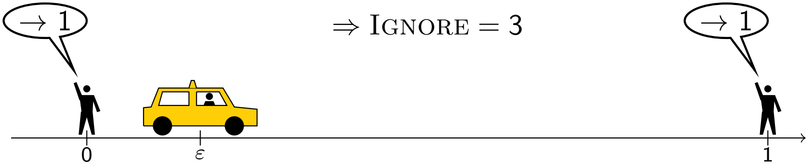
Example:

Some simple algorithms

IGNORE:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

Example:

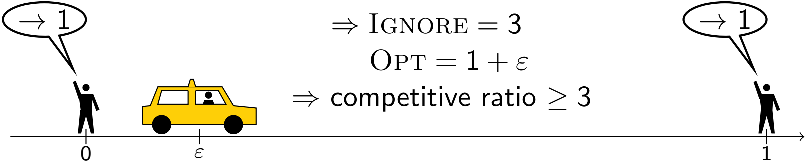


Some simple algorithms

IGNORE:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

Example:



$\Rightarrow$ IGNORE $= 3$

OPT $= 1 + \varepsilon$

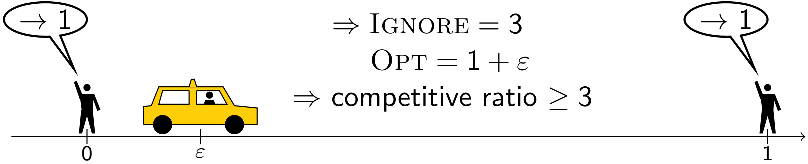
$\Rightarrow$ competitive ratio ≥ 3

Some simple algorithms

IGNORE:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

Example:



$\Rightarrow$ IGNORE $= 3$

OPT $= 1 + \varepsilon$

$\Rightarrow$ competitive ratio ≥ 3

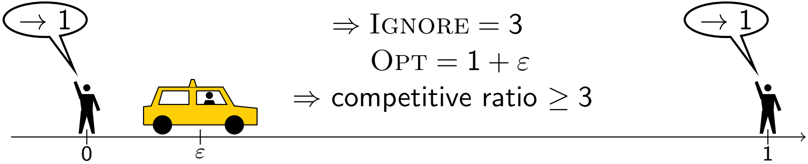
Theorem. The competitive ratio of IGNORE is 4. [Birx'20][Krumke'01]

Some simple algorithms

IGNORE:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

Example:



$\Rightarrow$ IGNORE $= 3$
OPT $= 1 + \varepsilon$
$\Rightarrow$ competitive ratio ≥ 3

Theorem. The competitive ratio of IGNORE is 4. [Birx'20][Krumke'01]

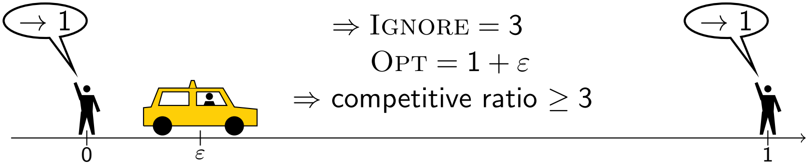
REPLAN: start optimal schedule whenever a request appears

Some simple algorithms

Ignore:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

Example:



$\Rightarrow$ Ignore $= 3$
Opt $= 1 + \varepsilon$
$\Rightarrow$ competitive ratio ≥ 3

Theorem. The competitive ratio of Ignore is 4. [Birx'20][Krumke'01]

Replan: start optimal schedule whenever a request appears

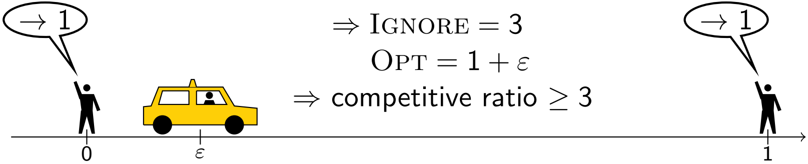
Theorem. The competitive ratio of Replan is in [2.5,4]. [Aussiello et al.'01][Birx'20]

Some simple algorithms

IGNORE:

- when idle: start optimal schedule over unserved requests
- never interrupt a schedule

Example:



$\Rightarrow$ IGNORE $= 3$
OPT $= 1 + \varepsilon$
$\Rightarrow$ competitive ratio ≥ 3

Theorem. The competitive ratio of IGNORE is 4. [Birx'20][Krumke'01]

REPLAN: start optimal schedule whenever a request appears

Theorem. The competitive ratio of REPLAN is in [2.5,4]. [Aussiello et al.'01][Birx'20]

$\to$ "interpolate" between IGNORE and REPLAN

The LAZY-algorithm

$\textsc{Lazy}_\alpha$
new request revealed:
▶ if possible before time $\alpha \cdot \textsc{Opt}(t)$: RESET
when idle:
▶ start schedule at time $t \geq \alpha \cdot \textsc{Opt}(t)$

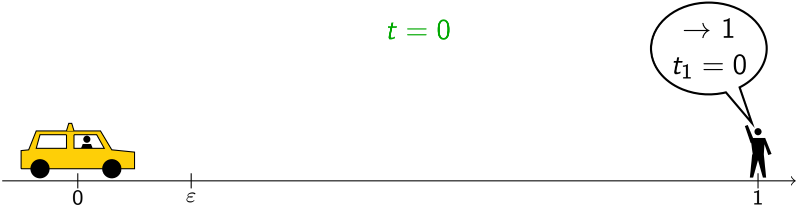
- $\textsc{Opt}(t)$: offline optimum of requests revealed before time t
- RESET: deliver loaded requests and return to origin

The Lazy-algorithm

$\textsc{Lazy}_\alpha$
new request revealed:
▶ if possible before time $\alpha \cdot \textsc{Opt}(t)$: $\textsc{reset}$
when idle:
▶ start schedule at time $t \geq \alpha \cdot \textsc{Opt}(t)$

- $\textsc{Opt}(t)$: offline optimum of requests revealed before time t
- $\textsc{reset}$: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):

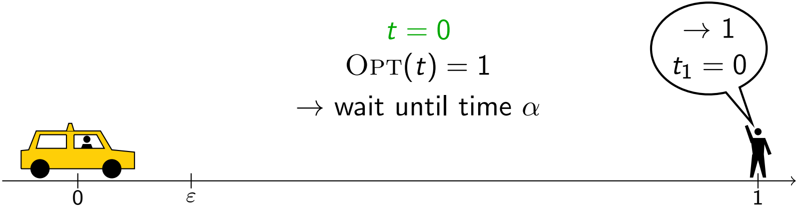


The LAZY-algorithm

LAZY$_\alpha$
new request revealed:
▶ if possible before time $\alpha \cdot \text{OPT}(t)$: RESET
when idle:
▶ start schedule at time $t \geq \alpha \cdot \text{OPT}(t)$

- $\text{OPT}(t)$: offline optimum of requests revealed before time t
- RESET: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):



The LAZY-algorithm

LAZY$_\alpha$

new request revealed:

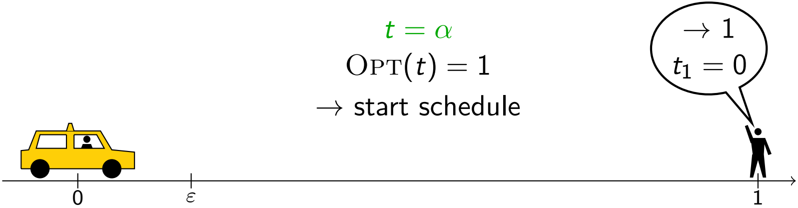
- if possible before time $\alpha \cdot \text{OPT}(t)$: RESET

when idle:

- start schedule at time $t \geq \alpha \cdot \text{OPT}(t)$

- $\text{OPT}(t)$: offline optimum of requests revealed before time t
- RESET: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):

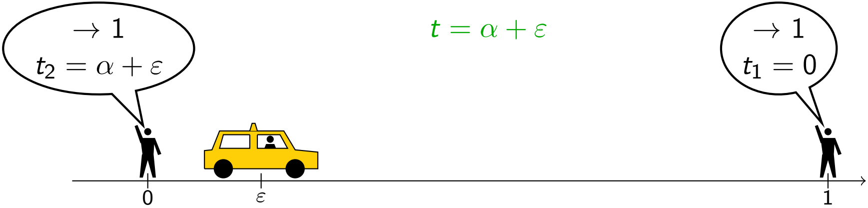


The LAZY-algorithm

LAZY_α
new request revealed: ▶ if possible before time $\alpha \cdot \text{OPT}(t)$: RESET when idle: ▶ start schedule at time $t \geq \alpha \cdot \text{OPT}(t)$

- $\text{OPT}(t)$: offline optimum of requests revealed before time t
- RESET: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):

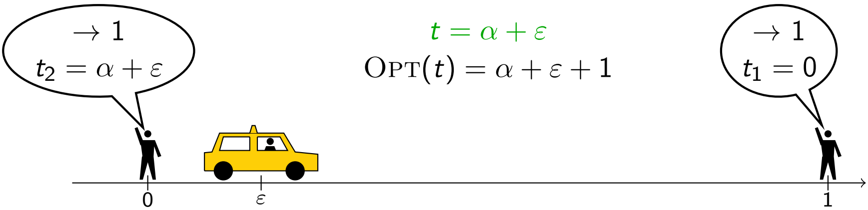


The LAZY-algorithm

LAZY$_\alpha$
new request revealed:
▶ if possible before time $\alpha \cdot \text{OPT}(t)$: RESET
when idle:
▶ start schedule at time $t \geq \alpha \cdot \text{OPT}(t)$

- $\text{OPT}(t)$: offline optimum of requests revealed before time t
- RESET: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):

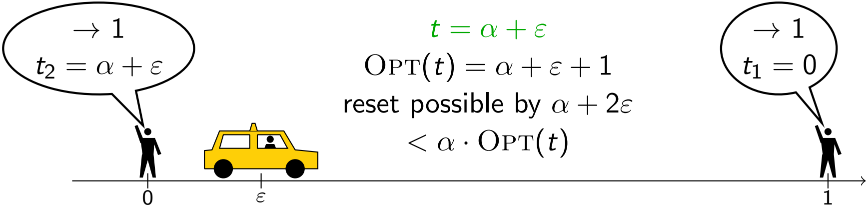


The LAZY-algorithm

LAZY$_\alpha$
new request revealed:
▸ if possible before time $\alpha \cdot \text{OPT}(t)$: RESET
when idle:
▸ start schedule at time $t \geq \alpha \cdot \text{OPT}(t)$

- $\text{OPT}(t)$: offline optimum of requests revealed before time t
- RESET: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):



The $\textsc{Lazy}$-algorithm

$\textsc{Lazy}_\alpha$

new request revealed:

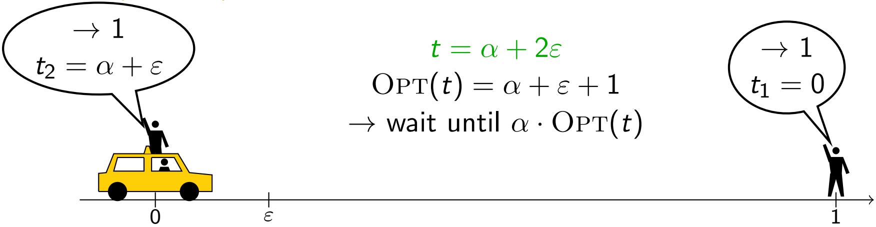
- if possible before time $\alpha \cdot \textsc{Opt}(t)$: $\textsc{reset}$

when idle:

- start schedule at time $t \geq \alpha \cdot \textsc{Opt}(t)$

- $\textsc{Opt}(t)$: offline optimum of requests revealed before time t
- $\textsc{reset}$: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):

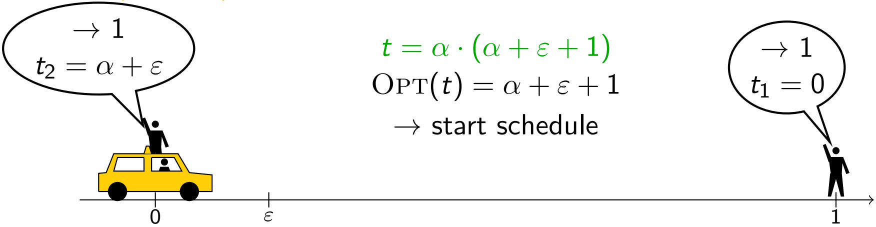


The Lazy-algorithm

Lazy$_\alpha$
new request revealed: ▶ if possible before time $\alpha \cdot \mathrm{OPT}(t)$: RESET when idle: ▶ start schedule at time $t \geq \alpha \cdot \mathrm{OPT}(t)$

- $\mathrm{OPT}(t)$: offline optimum of requests revealed before time t
- RESET: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):



The LAZY-algorithm

LAZY_α
new request revealed:
▶ if possible before time $\alpha \cdot \text{OPT}(t)$: RESET
when idle:
▶ start schedule at time $t \geq \alpha \cdot \text{OPT}(t)$

- $\text{OPT}(t)$: offline optimum of requests revealed before time t
- RESET: deliver loaded requests and return to origin

Example ($\alpha = 1.5$):

$$\text{LAZY}_\alpha = \alpha \cdot (\alpha + \varepsilon + 1) + 1$$
$$\text{OPT} = \alpha + \varepsilon + 1$$

State of the art and our results

Theorem. $\textsc{Lazy}_\alpha$ is

a) 2.457-competitive on every metric space, for $\alpha = 1.457$.
b) 2.366-competitive on the half-line, for $\alpha = 1.366$.
c) There are no better choices for α.

State of the art and our results

Theorem. $\textsc{Lazy}_\alpha$ is

a) 2.457-competitive on every metric space, for $\alpha = 1.457$.
b) 2.366-competitive on the half-line, for $\alpha = 1.366$.
c) There are no better choices for α.

metric space	lower bound	old upper bound	new upper bound
general	2.05	2.618 [1]	**2.457**
line	2.05 [2]	2.618	2.457
half-line	1.9 [3]	2.618	**2.366**

[1] B., Disser, Mosis, Weckbecker (2022)
[2] Birx, Disser, Schewior (2022)
[3] Lipmann (2003)

State of the art and our results

Theorem. $\textsc{Lazy}_\alpha$ is

a) 2.457-competitive on every metric space, for $\alpha = 1.457$.
b) 2.366-competitive on the half-line, for $\alpha = 1.366$.
c) There are no better choices for α.

metric space	lower bound	old upper bound	new upper bound
general	2.05	2.618 [1]	**2.457**
line	2.05 [2]	2.618	2.457
half-line	1.9 [3]	2.618	**2.366**

$\rightarrow$ key to analysis of $\textsc{Lazy}$: factor-revealing approach

[1] B., Disser, Mosis, Weckbecker (2022)
[2] Birx, Disser, Schewior (2022)
[3] Lipmann (2003)

Analysis of LAZY

LAZY_α
new request revealed:
▶ if possible before $\alpha\text{OPT}(t)$: RESET
when idle:
▶ start schedule at $t \geq \alpha \cdot \text{OPT}(t)$

Analysis of $\textsc{Lazy}$

Thm. For $\alpha = 1.457$,
$\textsc{Lazy}_\alpha(\sigma) \leq (1+\alpha)\textsc{Opt}(\sigma)\ \forall\sigma$

$\textsc{Lazy}_\alpha$

new request revealed:

- if possible before $\alpha\textsc{Opt}(t)$: $\textsc{reset}$

when idle:

- start schedule at $t \geq \alpha \cdot \textsc{Opt}(t)$

Analysis of Lazy

Thm. For $\alpha = 1.457$,
$\textsc{Lazy}_\alpha(\sigma) \leq (1+\alpha)\textsc{Opt}(\sigma)\ \forall\sigma$

$\textsc{Lazy}_\alpha$
new request revealed:
▶ if possible before $\alpha\textsc{Opt}(t)$: RESET
when idle:
▶ start schedule at $t \geq \alpha \cdot \textsc{Opt}(t)$

1st step: define suitable variables

Analysis of Lazy

Thm. For $\alpha = 1.457$,
$\textsc{Lazy}_\alpha(\sigma) \leq (1+\alpha)\textsc{Opt}(\sigma)\ \forall\sigma$

$\textsc{Lazy}_\alpha$

new request revealed:
- if possible before $\alpha\textsc{Opt}(t)$: RESET

when idle:
- start schedule at $t \geq \alpha \cdot \textsc{Opt}(t)$

1st step: define suitable variables

- t^*: start time of last schedule
- s: duration of last schedule

Analysis of $\textsc{Lazy}$

$\textsc{Lazy}_\alpha$
new request revealed:
▶ if possible before $\alpha\textsc{Opt}(t)$: $\textsc{reset}$
when idle:
▶ start schedule at $t \geq \alpha \cdot \textsc{Opt}(t)$

Thm. For $\alpha = 1.457$,
$\textsc{Lazy}_\alpha(\sigma) \leq (1+\alpha)\textsc{Opt}(\sigma)\ \forall\sigma$

1st step: define suitable variables

- t^*: start time of last schedule
- s: duration of last schedule

$\Rightarrow \textsc{Lazy}_\alpha(\sigma) = t^* + s$

Analysis of $\textsc{Lazy}$

Thm. For $\alpha = 1.457$,
$\textsc{Lazy}_\alpha(\sigma) \leq (1+\alpha)\textsc{Opt}(\sigma)\ \forall\sigma$

$\textsc{Lazy}_\alpha$

new request revealed:

- if possible before $\alpha\textsc{Opt}(t)$: RESET

when idle:

- start schedule at $t \geq \alpha \cdot \textsc{Opt}(t)$

1st step: define suitable variables

- t^*: start time of last schedule
- s: duration of last schedule

$\Rightarrow \textsc{Lazy}_\alpha(\sigma) = t^* + s$

2nd step: find inequalities

- $t^* \geq \alpha \cdot \textsc{Opt}$, ...

Analysis of $\textsc{Lazy}$

$\textsc{Lazy}_\alpha$

new request revealed:

- if possible before $\alpha\textsc{Opt}(t)$: RESET

when idle:

- start schedule at $t \geq \alpha \cdot \textsc{Opt}(t)$

Thm. For $\alpha = 1.457$,
$\textsc{Lazy}_\alpha(\sigma) \leq (1+\alpha)\textsc{Opt}(\sigma)\ \forall\sigma$

1st step: define suitable variables

- t^*: start time of last schedule
- s: duration of last schedule

$\Rightarrow \textsc{Lazy}_\alpha(\sigma) = t^* + s$

2nd step: find inequalities

- $t^* \geq \alpha \cdot \textsc{Opt}$, ...

Question: Is Thm implied?

Analysis of Lazy

Thm. For $\alpha = 1.457$,
$\text{Lazy}_\alpha(\sigma) \leq (1+\alpha)\text{Opt}(\sigma)\ \forall\sigma$

1st step: define suitable variables

- t^*: start time of last schedule
- s: duration of last schedule

$\Rightarrow \text{Lazy}_\alpha(\sigma) = t^* + s$

2nd step: find inequalities

- $t^* \geq \alpha \cdot \text{Opt}$, ...

Question: Is Thm implied?

Lazy_α
new request revealed: ▶ if possible before $\alpha\text{Opt}(t)$: RESET when idle: ▶ start schedule at $t \geq \alpha \cdot \text{Opt}(t)$

- solve "adversary problem"

$$\max \frac{t^* + s}{\text{Opt}}$$
$$\text{s.t. } t^* \geq \alpha \cdot \text{Opt}$$
$$\vdots$$

Analysis of $\textsc{Lazy}$

Thm. For $\alpha = 1.457$,
$\textsc{Lazy}_\alpha(\sigma) \leq (1+\alpha)\textsc{Opt}(\sigma)\ \forall\sigma$

1st step: define suitable variables

- t^*: start time of last schedule
- s: duration of last schedule

$\Rightarrow \textsc{Lazy}_\alpha(\sigma) = t^* + s$

2nd step: find inequalities

- $t^* \geq \alpha \cdot \textsc{Opt}$, ...

Question: Is Thm implied?

$\textsc{Lazy}_\alpha$
new request revealed:
▶ if possible before $\alpha\textsc{Opt}(t)$: RESET
when idle:
▶ start schedule at $t \geq \alpha \cdot \textsc{Opt}(t)$

- solve "adversary problem"

$$\max \frac{t^* + s}{\textsc{Opt}}$$
$$\text{s.t. } t^* \geq \alpha \cdot \textsc{Opt}$$
$$\vdots$$

- rescaling σ gives

$$\text{(ALP)} \quad \max\ t^* + s$$
$$\text{s.t. } \textsc{Opt} = 1$$
$$t^* \geq \alpha \cdot \textsc{Opt}$$
$$\vdots$$

Analysis of LAZY

Thm. For $\alpha = 1.457$,
$\text{LAZY}_\alpha(\sigma) \leq (1+\alpha)\text{OPT}(\sigma)\ \forall\sigma$

LAZY_α
new request revealed: ▶ if possible before $\alpha\text{OPT}(t)$: RESET when idle: ▶ start schedule at $t \geq \alpha \cdot \text{OPT}(t)$

1st step: define suitable variables

- t^*: start time of last schedule
- s: duration of last schedule

$\Rightarrow \text{LAZY}_\alpha(\sigma) = t^* + s$

2nd step: find inequalities

- $t^* \geq \alpha \cdot \text{OPT}$, ...

Question: Is Thm implied?

- solve "adversary problem"

$$\max \frac{t^* + s}{\text{OPT}}$$
$$\text{s.t. } t^* \geq \alpha \cdot \text{OPT}$$
$$\vdots$$

- rescaling σ gives

$$\text{(ALP)} \quad \max\ t^* + s$$
$$\text{s.t. } \text{OPT} = 1$$
$$t^* \geq \alpha \cdot \text{OPT}$$
$$\vdots$$

- solution of (ALP): $\alpha + 5$ $(t^* = \alpha, s = 5)$

Analysis of $\textsc{Lazy}$

Thm. For $\alpha = 1.457$,
$\textsc{Lazy}_\alpha(\sigma) \leq (1+\alpha)\textsc{Opt}(\sigma)\ \forall \sigma$

$\textsc{Lazy}_\alpha$
new request revealed:
- if possible before $\alpha\textsc{Opt}(t)$: RESET

when idle:
- start schedule at $t \geq \alpha \cdot \textsc{Opt}(t)$

1st step: define suitable variables
- t^*: start time of last schedule
- s: duration of last schedule

$\Rightarrow \textsc{Lazy}_\alpha(\sigma) = t^* + s$

2nd step: find inequalities
- $t^* \geq \alpha \cdot \textsc{Opt}$, ...

Question: Is Thm implied?

- solve "adversary problem"

$$\max \frac{t^* + s}{\textsc{Opt}}$$
$$\text{s.t. } t^* \geq \alpha \cdot \textsc{Opt}$$
$$\vdots$$

- rescaling σ gives

$$\text{(ALP)}\ \max t^* + s$$
$$\text{s.t. } \textsc{Opt} = 1$$
$$t^* \geq \alpha \cdot \textsc{Opt}$$
$$\vdots$$

- solution of (ALP): $\alpha + 5$ $(t^* = \alpha, s = 5)$

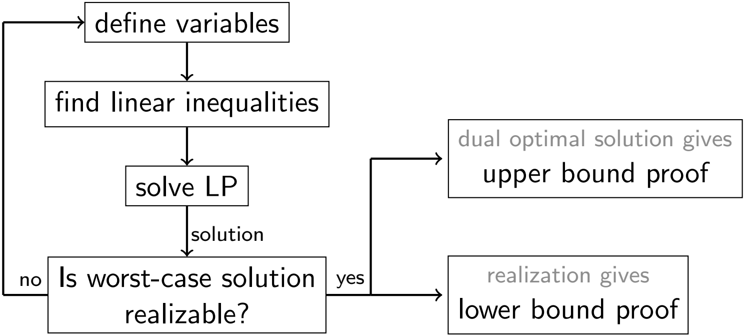
$\to$ add inequality $s \leq 2 \cdot \textsc{Opt}$

Factor-revealing approach [Bienkowski, Kraska, Liu (2021)]

- useful to assemble linear inequalities (for analysis of a fixed algorithm)

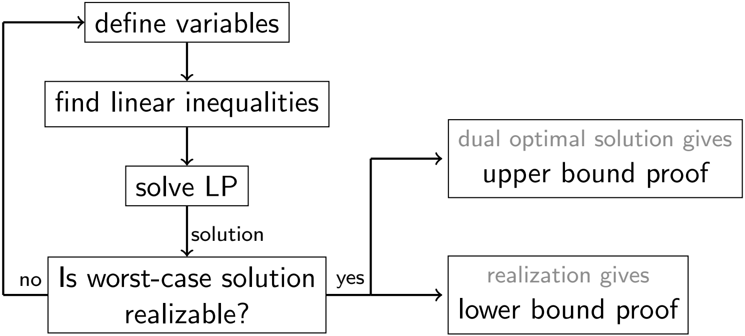
Factor-revealing approach [Bienkowski, Kraska, Liu (2021)]

- useful to assemble linear inequalities (for analysis of a fixed algorithm)



Factor-revealing approach [Bienkowski, Kraska, Liu (2021)]

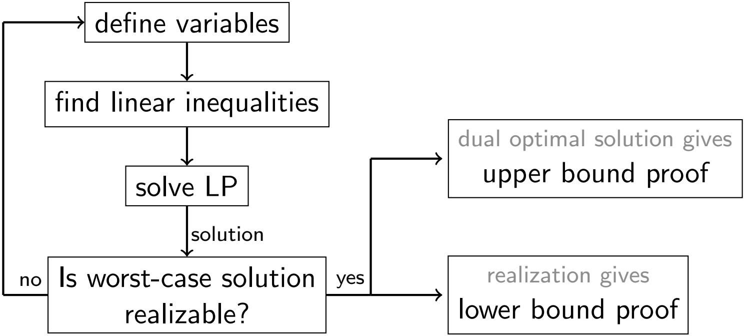
- useful to assemble linear inequalities (for analysis of a fixed algorithm)



- our solution requires discrete variables

Factor-revealing approach [Bienkowski, Kraska, Liu (2021)]

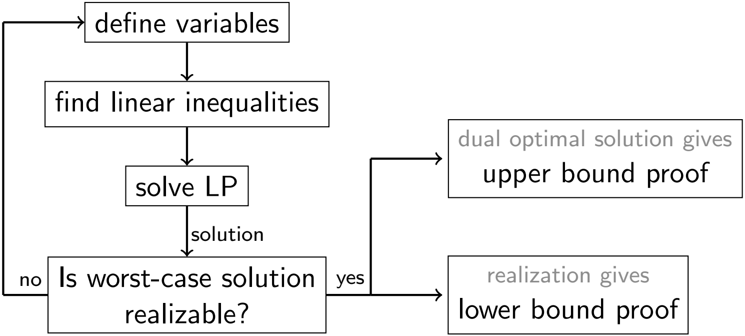
- useful to assemble linear inequalities (for analysis of a fixed algorithm)



- our solution requires discrete variables
  $\rightarrow$ need dual solution for every Branch & Bound node

Factor-revealing approach [Bienkowski, Kraska, Liu (2021)]

- useful to assemble linear inequalities (for analysis of a fixed algorithm)



- our solution requires discrete variables
  → need dual solution for every Branch & Bound node
- our work: purely analytic proof informed by factor-revealing

Takeaways

- dial-a-ride problem: serve transportation requests appearing over time
  - minimize completion time

Takeaways

- dial-a-ride problem: serve transportation requests appearing over time
  - minimize completion time
- competitive analysis: compare performance to offline optimum

Takeaways

- dial-a-ride problem: serve transportation requests appearing over time
  - minimize completion time
- competitive analysis: compare performance to offline optimum
- IGNORE is 4-competitive and REPLAN is $\geq$ 2.5-competitive

Takeaways

- dial-a-ride problem: serve transportation requests appearing over time
  - minimize completion time
- competitive analysis: compare performance to offline optimum
- IGNORE is 4-competitive and REPLAN is $\geq$ 2.5-competitive

Theorem. LAZY is
- 2.457-competitive on general metric spaces,
- 2.366-competitive on the half-line.

Takeaways

- dial-a-ride problem: serve transportation requests appearing over time
  - minimize completion time
- competitive analysis: compare performance to offline optimum
- IGNORE is 4-competitive and REPLAN is $\geq$ 2.5-competitive

Theorem. LAZY is
- 2.457-competitive on general metric spaces,
- 2.366-competitive on the half-line.

- key to analysis: factor-revealing approach
  - method to assemble linear inequalities