Rate-Monotonic Schedulability of Implicit-Deadline Tasks is NP-hard Beyond Liu and Layland's Bound

Pontus Ekberg

Uppsala University

RTSS 2020

Background

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-hard Exponential time algorithm exists
Utilization bounded by a constant $c < 1$	Polynomial time with RM priorities and $c \leqslant \ln(2)$	Weakly NP-complete for $c > 0$ Pseudo-polynomial time algorithm exists	Weakly NP-hard for $c > 0$ Pseudo-polynomial time algorithm exists

Background

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-hard Exponential time algorithm exists
Utilization bounded by a constant $c < 1$	Polynomial time with RM priorities and $c \leqslant \ln(2)$	Weakly NP-complete for $c > 0$ Pseudo-polynomial time algorithm exists	Weakly NP-hard for $c > 0$ Pseudo-polynomial time algorithm exists

Background

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-hard Exponential time algorithm exists
Utilization bounded by a constant $c < 1$	Polynomial time with RM priorities and $c \leqslant \ln(2)$	Weakly NP-complete for $c > 0$ Pseudo-polynomial time algorithm exists	Weakly NP-hard for $c > 0$ Pseudo-polynomial time algorithm exists

Background

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-hard Exponential time algorithm exists
Utilization bounded by a constant $c < 1$	Polynomial time with RM priorities and $c \leqslant \ln(2)$	Weakly NP-complete for $c > 0$ Pseudo-polynomial time algorithm exists	Weakly NP-hard for $c > 0$ Pseudo-polynomial time algorithm exists

Background

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-hard Exponential time algorithm exists
Utilization bounded by a constant $c < 1$	Polynomial time with RM priorities and $c \leqslant \ln(2)$	Weakly NP-complete for $c > 0$ Pseudo-polynomial time algorithm exists	Weakly NP-hard for $c > 0$ Pseudo-polynomial time algorithm exists

What if $c > \ln(2)$ or if the priorities are non-RM?

Complexity of bounded FP-schedulability

Conjecture by Rothvoß, 2009

RM-schedulability of task sets with implicit deadlines and utilization bounded by constant c is in P for all $c < 1$.

Complexity of bounded FP-schedulability

Conjecture by Rothvoß, 2009

RM-schedulability of task sets with implicit deadlines and utilization bounded by constant c is in P for all $c < 1$.

New results

FP-schedulability is NP-complete even if restricted to

1. RM priorities and $c > \ln(2)$, or
2. non-RM priorities and $c > 0$.

Complexity of bounded FP-schedulability

Conjecture by Rothvoß, 2009

RM-schedulability of task sets with implicit deadlines and utilization bounded by constant c is in P for all $c < 1$.

New results

FP-schedulability is NP-complete even if restricted to

1. RM priorities and $c > \ln(2)$, or
2. non-RM priorities and $c > 0$.

An added layer of abstraction

Goal

Show that RM-schedulability is NP-complete even when utilization is bounded by any constant $c > \ln(2)$.

An added layer of abstraction

Goal

Show that RM-schedulability is NP-complete even when utilization is bounded by any constant $c > \ln(2)$.

Hurdle

There is no minimum c, such that $c > \ln(2)$.

An added layer of abstraction

Goal

Show that RM-schedulability is NP-complete even when utilization is bounded by any constant $c > \ln(2)$.

Hurdle

There is no minimum c, such that $c > \ln(2)$.

$c = 0.7$?

An added layer of abstraction

Goal

Show that RM-schedulability is NP-complete even when utilization is bounded by any constant $c > \ln(2)$.

Hurdle

There is no minimum c, such that $c > \ln(2)$.

$c = 0.7?$ $\quad$ $c = 0.695?$

An added layer of abstraction

Goal

Show that RM-schedulability is NP-complete even when utilization is bounded by any constant $c > \ln(2)$.

Hurdle

There is no minimum c, such that $c > \ln(2)$.

$c = 0.7?$ $\quad$ $c = 0.695?$ $\quad$ $c = \ln(2) + \epsilon?$

An added layer of abstraction

Goal

Show that RM-schedulability is NP-complete even when utilization is bounded by any constant $c > \ln(2)$.

Hurdle

There is no minimum c, such that $c > \ln(2)$.

$c = 0.7?$ $\quad c = 0.695?$ $\quad c = \ln(2) + \epsilon?$

Approach

Show that for any $c > \ln(2)$ there *exists* a reduction from some NP-complete problem.

Classes of computational problems

An RM-schedulability class

- Utilization bounded by *some* constant $c > \ln(2)$
- Implicit deadlines

Classes of computational problems

An RM-schedulability class

- Utilization bounded by *some* constant $c > 0$
- Implicit deadlines for high-priority tasks
- Constrained deadline for the lowest-priority task

An RM-schedulability class

- Utilization bounded by *some* constant $c > \ln(2)$
- Implicit deadlines

All NP-complete
(Ekberg & Yi, 2017)

Classes of computational problems

An RM-schedulability class

- Utilization bounded by *some* constant $c > 0$
- Implicit deadlines for high-priority tasks
- Constrained deadline for the lowest-priority task

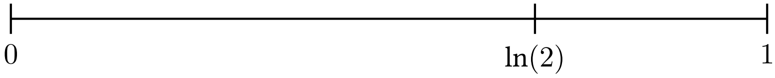
?

An RM-schedulability class

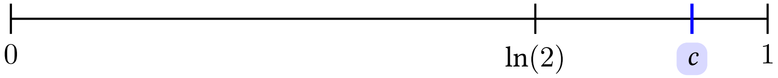
- Utilization bounded by *some* constant $c > \ln(2)$
- Implicit deadlines

All NP-complete
(Ekberg & Yi, 2017)

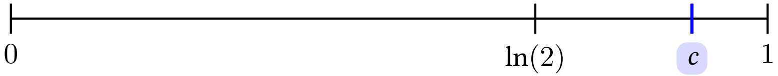
The existence of a reduction for a concrete c



The existence of a reduction for a concrete c



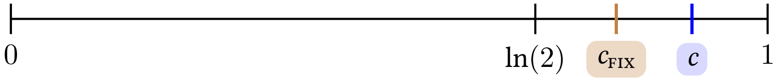
The existence of a reduction for a concrete c



RM-schedulability

- Utilization bounded by c
- Implicit deadlines

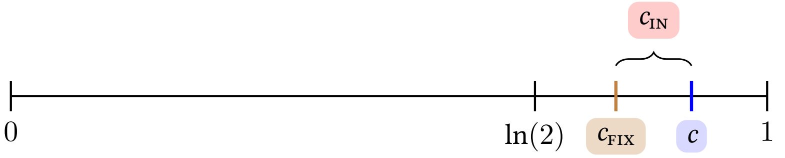
The existence of a reduction for a concrete c



RM-schedulability

- Utilization bounded by c
- Implicit deadlines

The existence of a reduction for a concrete c



RM-schedulability

- Utilization bounded by c
- Implicit deadlines

The existence of a reduction for a concrete c

RM-schedulability

- Utilization bounded by c_{IN}
- Implicit deadlines for high-priority tasks
- Constrained deadline for the lowest-priority task

RM-schedulability

- Utilization bounded by c
- Implicit deadlines

Back to the '70s

Back to the '70s

Corollary from Liu & Layland, 1973

For every $c > \ln(2)$, there exists an implicit-deadline task set with utilization $< c$ that *fully utilizes*✽ the processor.

Back to the '70s

Corollary from Liu & Layland, 1973

For every $c > \ln(2)$, there exists an implicit-deadline task set with utilization $< c$ that *fully utilizes*✿ the processor.

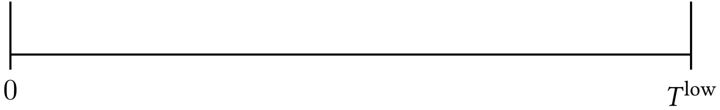
✿ It is schedulable, but no WCET can be increased.

Back to the '70s

Corollary from Liu & Layland, 1973

For every $c > \ln(2)$, there exists an implicit-deadline task set with utilization $< c$ that *fully utilizes*✿ the processor.

✿ It is schedulable, but no WCET can be increased.

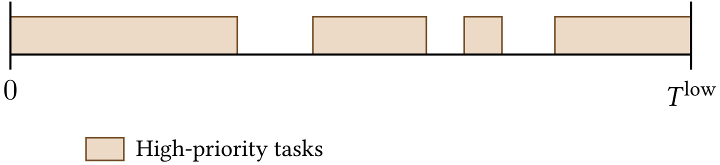


Back to the '70s

Corollary from Liu & Layland, 1973

For every $c > \ln(2)$, there exists an implicit-deadline task set with utilization $< c$ that *fully utilizes*✿ the processor.

✿ It is schedulable, but no WCET can be increased.

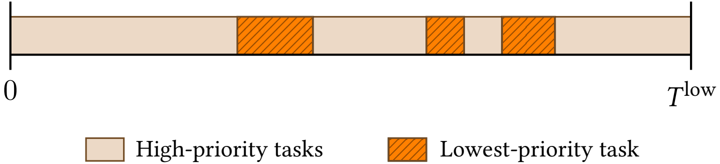


Back to the '70s

Corollary from Liu & Layland, 1973

For every $c > \ln(2)$, there exists an implicit-deadline task set with utilization $< c$ that *fully utilizes*✿ the processor.

✿ It is schedulable, but no WCET can be increased.



The existence of a reduction for a concrete c

RM-schedulability

- Utilization bounded by c_{IN}
- Implicit deadlines for high-priority tasks
- Constrained deadline for the lowest-priority task

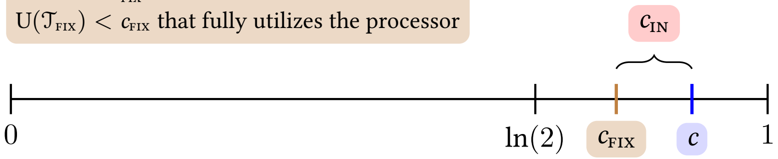
$\rightsquigarrow$

RM-schedulability

- Utilization bounded by c
- Implicit deadlines

THE EXISTENCE OF A REDUCTION FOR A CONCRETE c

Let $\mathcal{T}_{\text{FIX}}$ be a *constant* task set with $U(\mathcal{T}_{\text{FIX}}) < c_{\text{FIX}}$ that fully utilizes the processor



RM-SCHEDULABILITY

- Utilization bounded by c_{IN}
- Implicit deadlines for high-priority tasks
- Constrained deadline for the lowest-priority task

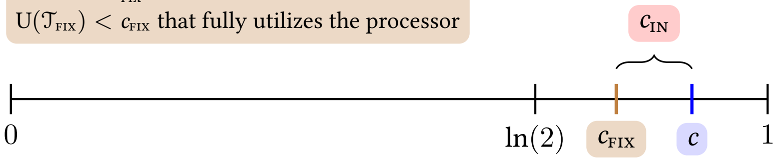
$\rightsquigarrow$

RM-SCHEDULABILITY

- Utilization bounded by c
- Implicit deadlines

The existence of a reduction for a concrete c

Let $\mathcal{T}_{\text{FIX}}$ be a *constant* task set with $U(\mathcal{T}_{\text{FIX}}) < c_{\text{FIX}}$ that fully utilizes the processor



RM-schedulability

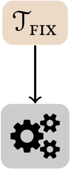
- Utilization bounded by c_{IN}
- Implicit deadlines for high-priority tasks
- Constrained deadline for the lowest-priority task

$\rightsquigarrow \mathcal{T}_{\text{FIX}} \rightsquigarrow$

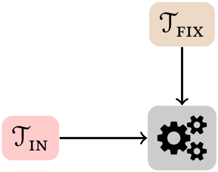
RM-schedulability

- Utilization bounded by c
- Implicit deadlines

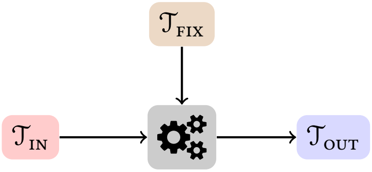
How the reduction works



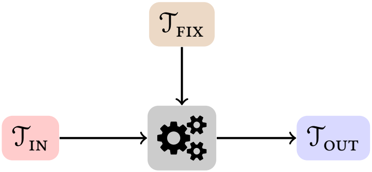
How the reduction works

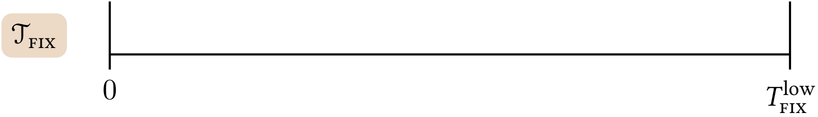


How the reduction works

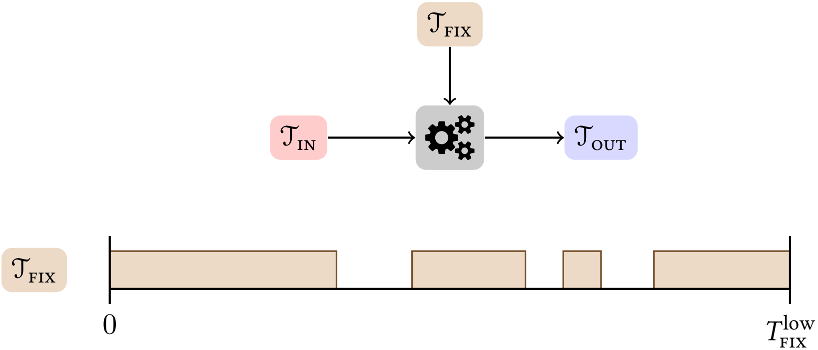


How the reduction works

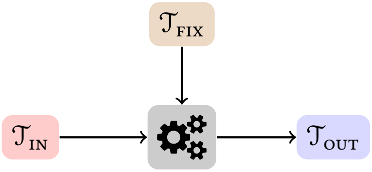


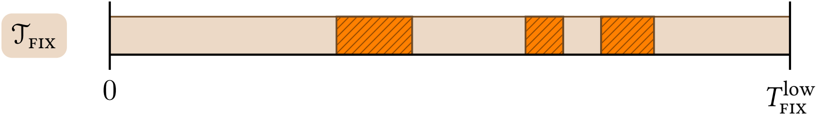


How the reduction works

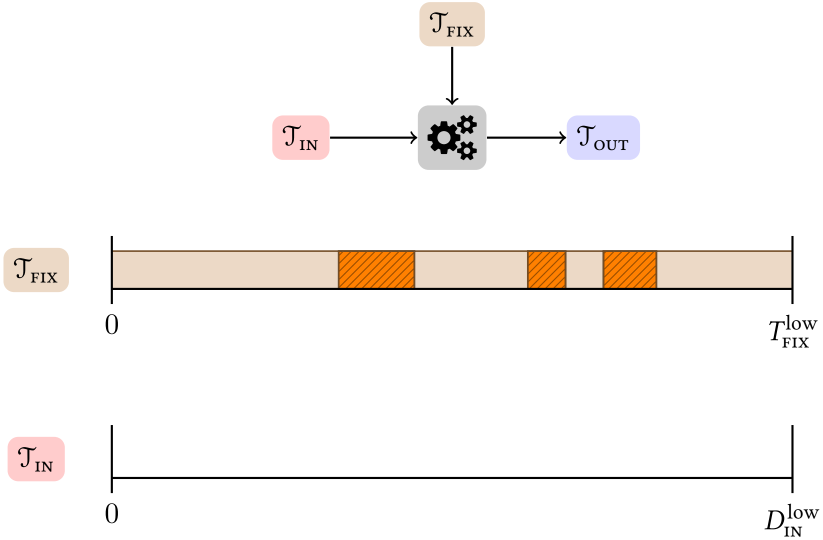


How the reduction works

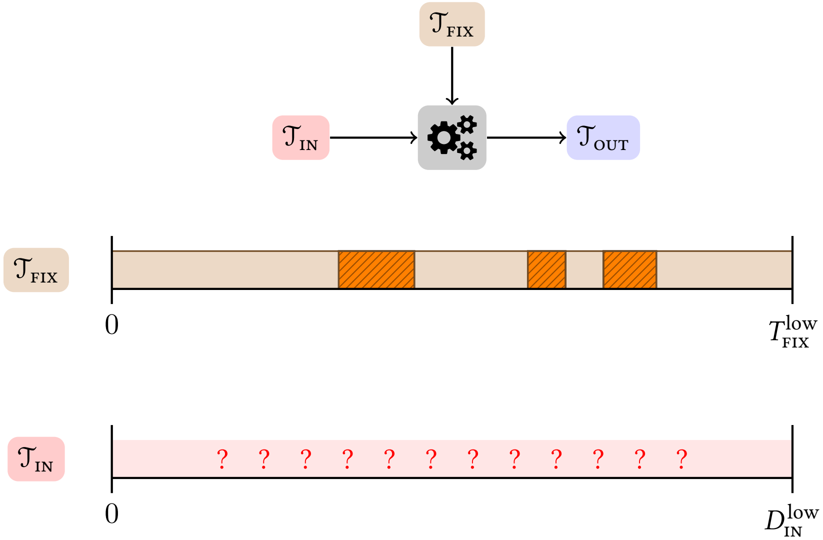




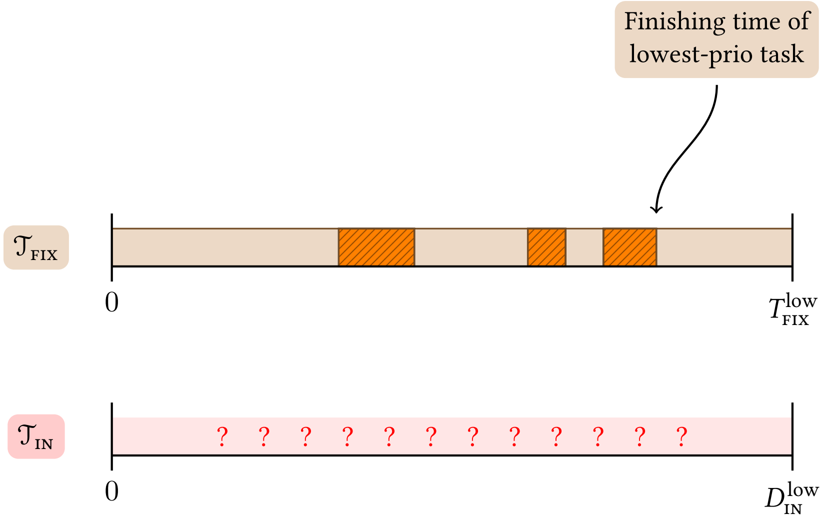
How the reduction works



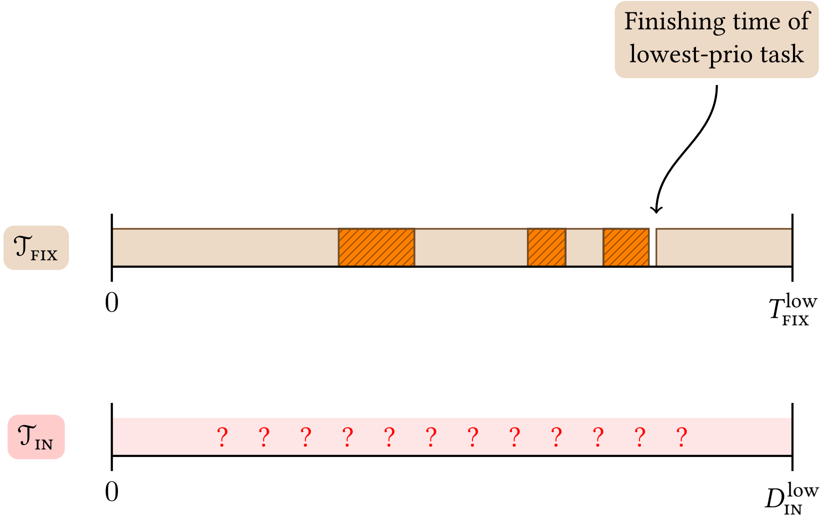
How the reduction works



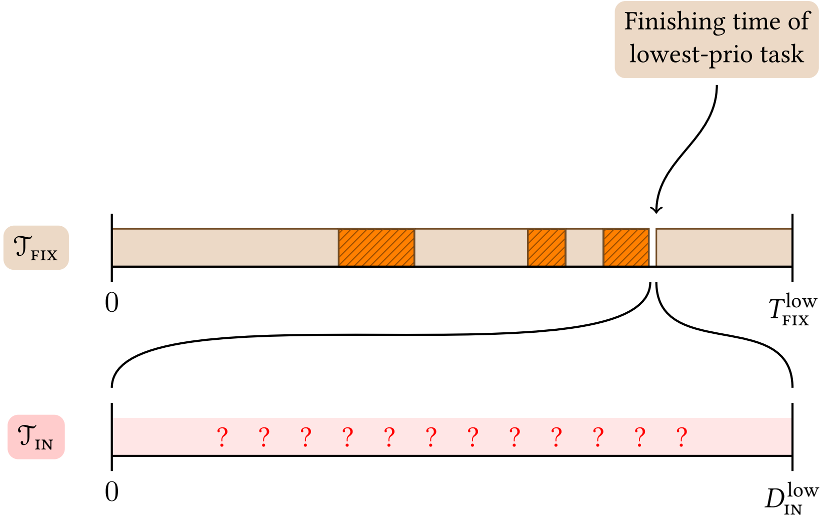
How the reduction works



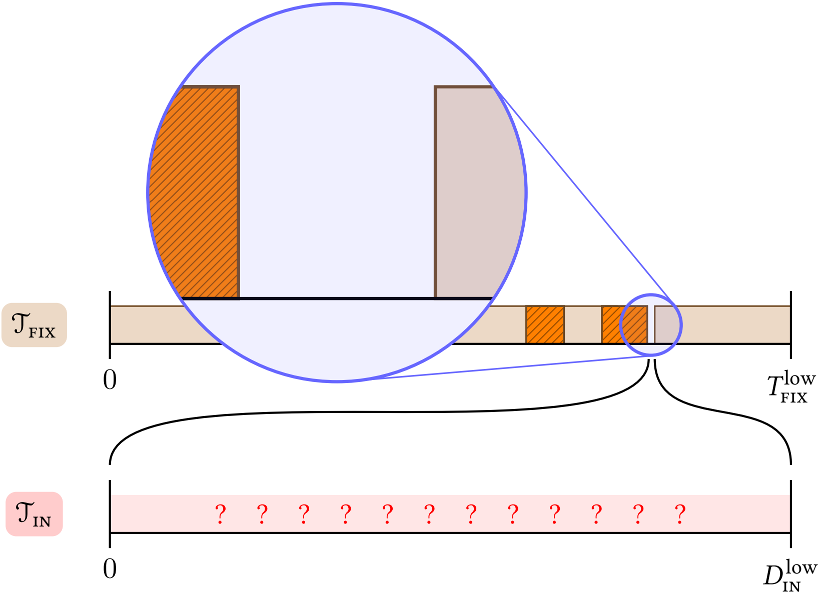
How the reduction works



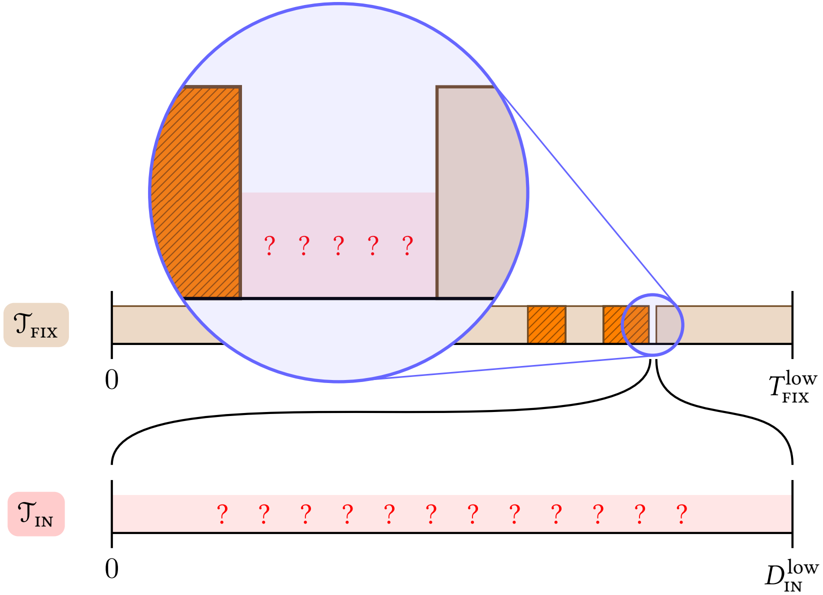
How the reduction works



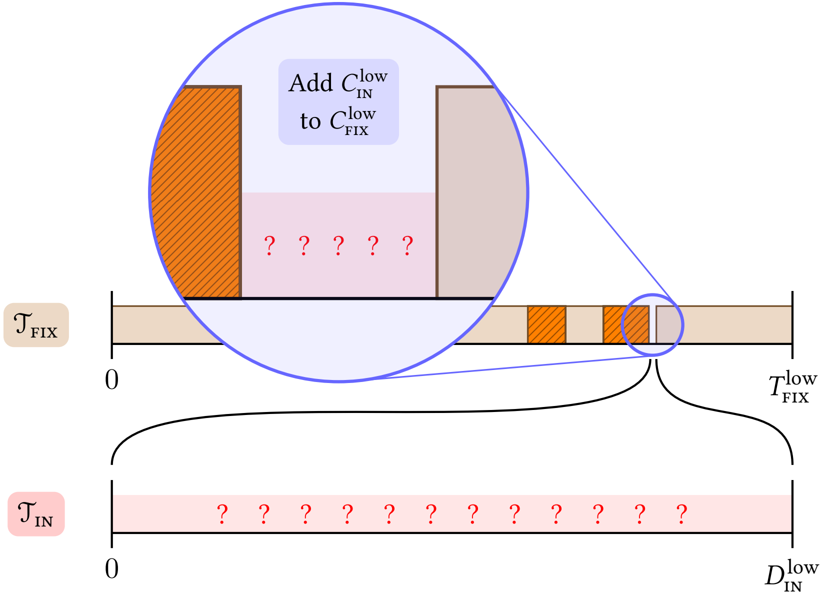
How the reduction works



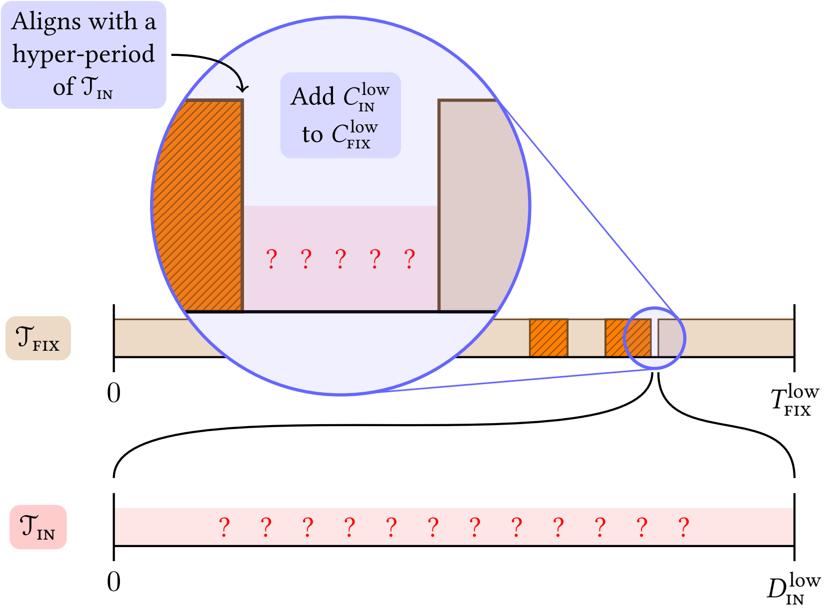
How the reduction works



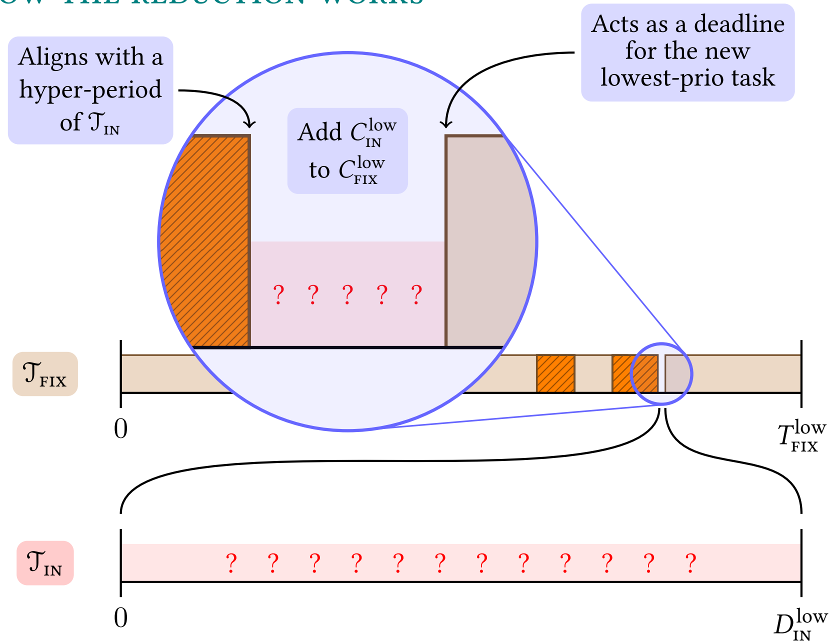
How the reduction works



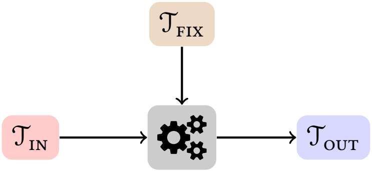
How the reduction works



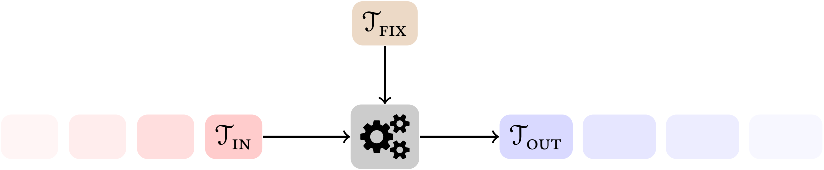
How the reduction works



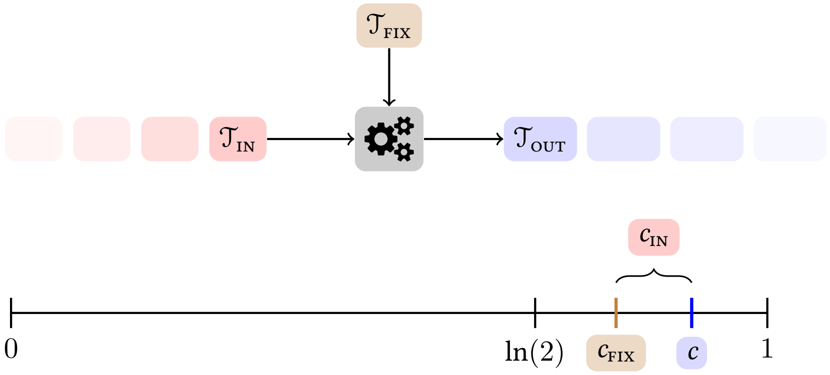
How the reduction works



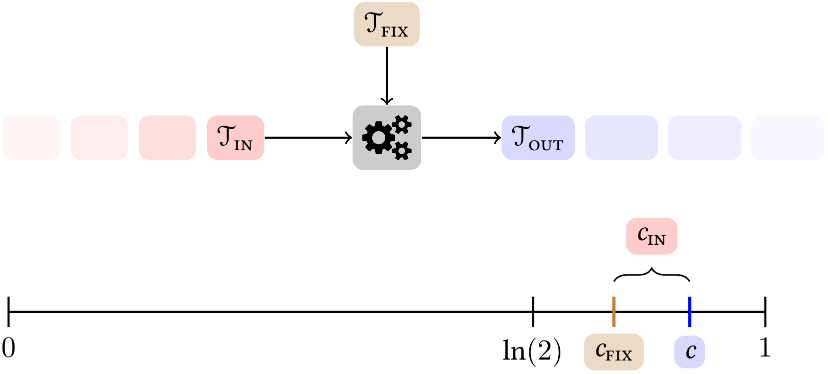
How the reduction works



How the reduction works

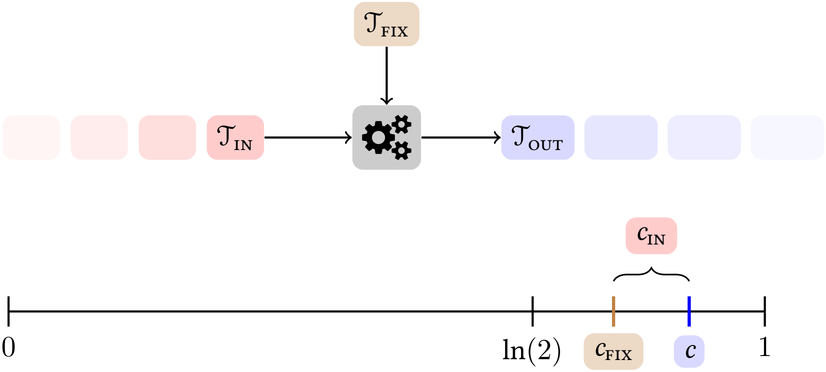


How the reduction works



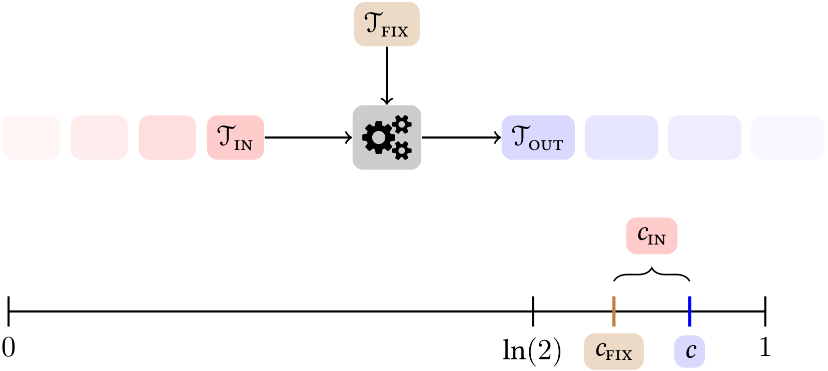
$$\mathrm{U}(\mathcal{T}_{\text{OUT}}) \leqslant \mathrm{U}(\mathcal{T}_{\text{IN}}) + \mathrm{U}(\mathcal{T}_{\text{FIX}})$$

How the reduction works



$$
\begin{aligned}
U(\mathcal{T}_{\text{OUT}}) &\leqslant U(\mathcal{T}_{\text{IN}}) + U(\mathcal{T}_{\text{FIX}}) \\
&\leqslant c_{\text{IN}} + c_{\text{FIX}}
\end{aligned}
$$

How the reduction works



$$
\begin{aligned}
\mathrm{U}(\mathcal{T}_{\text{OUT}}) &\leqslant \mathrm{U}(\mathcal{T}_{\text{IN}}) + \mathrm{U}(\mathcal{T}_{\text{FIX}}) \\
&\leqslant c_{\text{IN}} + c_{\text{FIX}} \\
&\leqslant c
\end{aligned}
$$

Conclusions

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-hard Exponential time algorithm exists
Utilization bounded by a constant $c < 1$	Polynomial time with RM priorities and $c \leqslant \ln(2)$	Weakly NP-complete for $c > 0$ Pseudo-polynomial time algorithm exists	Weakly NP-hard for $c > 0$ Pseudo-polynomial time algorithm exists

Conclusions

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-hard Exponential time algorithm exists
Utilization bounded by a constant $c < 1$	Weakly NP-complete with *(i)* RM and $c > \ln(2)$, or *(ii)* non-RM and $c > 0$ Otherwise in P	Weakly NP-complete for $c > 0$ Pseudo-polynomial time algorithm exists	Weakly NP-hard for $c > 0$ Pseudo-polynomial time algorithm exists

Conclusions

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-complete Pseudo-polynomial time algorithm exists	Weakly NP-hard Exponential time algorithm exists
Utilization bounded by a constant $c < 1$	Weakly NP-complete with *(i)* RM and $c > \ln(2)$, or *(ii)* non-RM and $c > 0$ Otherwise in P	Weakly NP-complete for $c > 0$ Pseudo-polynomial time algorithm exists	Weakly NP-hard for $c > 0$ Pseudo-polynomial time algorithm exists

$\forall$Thank you!

$\Diamond$

$\exists$Questions?