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

Pontus Ekberg

Uppsala University

RTSS 2020
## Background

<table>
<thead>
<tr>
<th></th>
<th>Implicit deadlines ((d = p))</th>
<th>Constrained deadlines ((d \leq p))</th>
<th>Arbitrary deadlines ((d, p) unrelated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary utilization</td>
<td>Weakly NP-complete</td>
<td>Weakly NP-complete</td>
<td>Weakly NP-hard</td>
</tr>
<tr>
<td></td>
<td>Pseudo-polynomial time algorithm exists</td>
<td>Pseudo-polynomial time algorithm exists</td>
<td>Exponential time algorithm exists</td>
</tr>
<tr>
<td>Utilization bounded by a constant (c &lt; 1)</td>
<td>Polynomial time with RM priorities and (c \leq \ln(2))</td>
<td>Weakly NP-complete for (c &gt; 0)</td>
<td>Weakly NP-hard for (c &gt; 0)</td>
</tr>
<tr>
<td></td>
<td>Pseudo-polynomial time algorithm exists</td>
<td>Pseudo-polynomial time algorithm exists</td>
<td>Pseudo-polynomial time algorithm exists</td>
</tr>
</tbody>
</table>
## Background

<table>
<thead>
<tr>
<th>Implicit deadlines ((d = p))</th>
<th>Constrained deadlines ((d \leq p))</th>
<th>Arbitrary deadlines ((d, p) unrelated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weakly NP-complete Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-complete Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-hard Exponential time algorithm exists</td>
</tr>
<tr>
<td>Polynomial time with RM priorities and (c \leq \ln(2))</td>
<td>Weakly NP-complete for (c &gt; 0) Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-hard for (c &gt; 0) Pseudo-polynomial time algorithm exists</td>
</tr>
</tbody>
</table>

**Arbitrary utilization**

**Utilization bounded by a constant \(c < 1\)**
## Background

<table>
<thead>
<tr>
<th>Utilization Type</th>
<th>Implicit deadlines ((d = p))</th>
<th>Constrained deadlines ((d \leq p))</th>
<th>Arbitrary deadlines ((d, p) unrelated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary utilization</td>
<td>Weakly NP-complete Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-complete Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-hard Exponential time algorithm exists</td>
</tr>
<tr>
<td>Utilization bounded by a constant (c &lt; 1)</td>
<td>Polynomial time with RM priorities and (c \leq \ln(2))</td>
<td>Weakly NP-complete for (c &gt; 0) Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-hard for (c &gt; 0) Pseudo-polynomial time algorithm exists</td>
</tr>
</tbody>
</table>
## Background

<table>
<thead>
<tr>
<th>Utilization</th>
<th>Implicit deadlines ( (d = p) )</th>
<th>Constrained deadlines ( (d \leq p) )</th>
<th>Arbitrary deadlines ( (d, p \text{ unrelated}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary utilization</td>
<td>Weakly NP-complete Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-complete Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-hard Exponential time algorithm exists</td>
</tr>
<tr>
<td>Utilization bounded by a constant ( c &lt; 1 )</td>
<td>Polynomial time with RM priorities and ( c \leq \ln(2) )</td>
<td>Weakly NP-complete for ( c &gt; 0 ) Pseudo-polynomial time algorithm exists</td>
<td>Weakly NP-hard for ( c &gt; 0 ) Pseudo-polynomial time algorithm exists</td>
</tr>
</tbody>
</table>
# Background

<table>
<thead>
<tr>
<th></th>
<th>Implicit deadlines ((d = p))</th>
<th>Constrained deadlines ((d \leq p))</th>
<th>Arbitrary deadlines ((d, p) unrelated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary utilization</td>
<td>Weakly NP-complete</td>
<td>Weakly NP-complete</td>
<td>Weakly NP-hard</td>
</tr>
<tr>
<td></td>
<td>Pseudo-polynomial time algorithm exists</td>
<td>Pseudo-polynomial time algorithm exists</td>
<td>Exponential time algorithm exists</td>
</tr>
<tr>
<td>Utilization bounded by a constant (c &lt; 1)</td>
<td>Polynomial time with RM priorities and (c \leq \ln(2))</td>
<td>Weakly NP-complete for (c &gt; 0)</td>
<td>Weakly NP-hard for (c &gt; 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo-polynomial time algorithm exists</td>
<td>Pseudo-polynomial time algorithm exists</td>
</tr>
</tbody>
</table>

What if \(c > \ln(2)\) or if the priorities are non-RM?
Complexity of bounded FP-schedulability

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 $c > 0$. 

*Pontus Ekberg*
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

$c = 0.7$

$c = 0.695$

$c = \ln(2) + \epsilon$

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

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

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

**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

(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

All NP-complete
(Ekberg & Yi, 2017)

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)
The existence of a reduction for a concrete $c$

![Diagram showing the range of $0$ to $\ln(2)$ to $1$]
The existence of a reduction for a concrete $c$

\[0 \quad \ln(2) \quad c \quad 1\]
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

$0 \rightarrow \ln(2) \rightarrow c_{\text{IN}} \rightarrow c_{\text{FIX}} \rightarrow c \rightarrow 1$
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

---

$0 \quad \ln(2) \quad c_{fix} \quad c \quad 1$
Corollary from Liu & Layland, 1973:

It is schedulable, but no WCET can be increased.
For every $c > \ln(2)$, there exists an implicit-deadline task set with utilization $< c$ that fully utilizes* the processor.

*Corollary from Liu & Layland, 1973
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.
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.
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.
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 $c$

Let $T_{\text{fix}}$ be a constant task set with $U(T_{\text{fix}}) < c_{\text{fix}}$ that fully utilizes the processor.

**RM-schedulability**
- Utilization bounded by $c_{\text{IN}}$
- Implicit deadlines for high-priority tasks
- Constrained deadline for the lowest-priority task

**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 $c_{\text{IN}}$
- Implicit deadlines for high-priority tasks
- Constrained deadline for the lowest-priority task

### RM-schedulability
- Utilization bounded by $c$
- Implicit deadlines
The existence of a reduction for a concrete $c$

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

RM-schedulability

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

RM-schedulability

- Utilization bounded by $c$
- Implicit deadlines
How the reduction works

$T_{\text{fix}}$
How the reduction works

\[ T_{\text{fix}} \]

\[ T_{\text{IN}} \]

\[ \text{Aligns with a hyper-period of } T_{\text{in}} \]

\[ \text{Acts as a deadline for the new lowest-priority task} \]
How the reduction works

\[ T_{\text{IN}} \rightarrow T_{\text{FIX}} \rightarrow T_{\text{OUT}} \]
How the reduction works

\[ T_{\text{fix}} \]

\[ T_{\text{in}} \]

\[ T_{\text{out}} \]

\[ T_{\text{fix}} \]

\[ T_{\text{low}} \)

\[ 0 \]

\[ T_{\text{fix}} \]
How the reduction works

- $\mathcal{T}_{IN}$
- $\mathcal{T}_{FIX}$
- $\mathcal{T}_{OUT}$

Finally:

- $\mathcal{T}_{low}$
- $\mathcal{T}_{FIX}$
How the reduction works

The reduction works by fixing the task $T_{fix}$ into $T_{in}$, which then aligns with a hyper-period of $T_{in}$. This acts as a deadline for the new lowest-priority task $T_{low}$. The finishing time of the lowest-priority task is represented by $T_{fix}$. Pontus EKberg
How the reduction works

\[ T_{\text{IN}} \rightarrow T_{\text{FIX}} \rightarrow T_{\text{OUT}} \]

Finishing time of the lowest-priority task?

\[ T_{\text{low}} \]

Aligns with a hyper-period of \( T_{\text{in}} \)

Acts as a deadline for the new lowest-priority task.

\[ D_{\text{IN}}^{\text{low}} \]
How the reduction works

$\mathcal{T}_{\text{in}} \rightarrow \mathcal{T}_{\text{fix}} \rightarrow \mathcal{T}_{\text{out}}$

$D_{\text{in}}^{\text{low}} \rightarrow \mathcal{T}_{\text{fix}} \rightarrow \mathcal{T}_{\text{in}}$

$\mathcal{T}_{\text{fix}}$ is the finishing time of the lowest-priority task.

$D_{\text{in}}^{\text{low}}$ aligns with a hyper-period of $\mathcal{T}_{\text{in}}$ and acts as a deadline for the new lowest-priority task.
How the reduction works

$T_{\text{FIX}}$

$T_{\text{IN}}$

Finishing time of lowest-prio task

$T_{\text{FIX}}$

$D_{\text{IN}}$

Pontus Ekberg
How the reduction works

$T_{\text{fix}}$ to $T_{\text{out}}$

Finishing time of lowest-prio task

Aligns with a hyper-period of $T_{\text{fix}}$

Acts as a deadline for the new lowest-prio task

$T_{\text{low}}$

$D_{\text{IN}}$

Pontus Ekberg
How the reduction works

Finishing time of lowest-prio task

$T_{\text{FIX}}$

$T_{\text{IN}}$

$D_{\text{IN}}^{\text{low}}$

$T_{\text{FIX}}^{\text{low}}$

Pontus Ekberg
How the reduction works

$T_{\text{fix}}$ to $T_{\text{out}}$

$T_{\text{low}}$ finish of lowest-priority task

Add $C_{\text{low}}$ to $C_{\text{fix}}$

Aligns with a hyper-period of $T_{\text{in}}$

Acts as a deadline for the new lowest-priority task
How the reduction works

$T^\text{fix}$

$T^\text{in}$

$\tau^\text{fix}$

$\tau^\text{in}$

$D^\text{low}_\text{in}$

$D^\text{low}_\text{in}$

$\tau^\text{low}$

Finishing time of lowest-priority task
How the Reduction works

Add $C_{\text{in}}^{\text{low}}$ to $C_{\text{fix}}^{\text{low}}$

Aligns with a hyper-period of $T_{\text{fix}}$

Acts as a deadline for the new lowest-prio task

Pontus Ekberg
How the reduction works

Aligns with a hyper-period of $\mathcal{T}_{IN}$

Add $C^{\text{low}}_{IN}$ to $C^{\text{low}}_{\text{FIX}}$

Finishing time of lowest-priority task

Add $C^{\text{low}}_{IN}$ to $C^{\text{low}}_{\text{FIX}}$

Aligns with a hyper-period of $\mathcal{T}_{IN}$

Acts as a deadline for the new lowest-priority task

$\mathcal{T}^{\text{fix}}$

$0$

$\mathcal{T}^{\text{low}}$

$\mathcal{T}^{\text{fix}}$

$0$

$D^{\text{low}}_{IN}$

Pontus EKberg
How the reduction works

Aligns with a hyper-period of $T_{IN}$

Add $C_{low}^{in}$ to $C_{fix}^{low}$

Acts as a deadline for the new lowest-prio task

Aligns with a hyper-period of $T_{IN}$

Add $C_{low}^{in}$ to $C_{fix}^{low}$

Acts as a deadline for the new lowest-prio task

$T_{fix}$

$T_{low}^{fix}$

$T_{in}$

$D_{low}^{in}$
The reduction works as follows:

\[ T_{\text{in}} \xrightarrow{\mathcal{T}_\text{fix}} T_{\text{out}} \]

where \( T_{\text{in}} \) is the input, \( T_{\text{fix}} \) is the fixed component, and \( T_{\text{out}} \) is the output. The inequality is:

\[
U(T_{\text{out}}) \leq U(T_{\text{in}}) + U(T_{\text{fix}}) \leq c_{\text{in}} + c_{\text{fix}}
\]
How the reduction works

$$T_{\text{FIX}}$$

$$T_{\text{IN}}$$ → $$T_{\text{OUT}}$$

$$\ln(2)$$

$$c_{\text{fix}}$$

$$U(T_{\text{out}}) \leq U(T_{\text{in}}) + U(T_{\text{fix}}) \leq c_{\text{in}} + c_{\text{fix}}$$

Pontus Ekberg
How the reduction works

\[ U(T_{\text{out}}) \leq U(T_{\text{in}}) + U(T_{\text{fix}}) \leq c_{\text{in}} + c_{\text{fix}} \leq c \]
How the reduction works

\[ U(\mathcal{T}_{\text{OUT}}) \leq U(\mathcal{T}_{\text{IN}}) + U(\mathcal{T}_{\text{FIX}}) \]
How the reduction works

\[ U(T_{OUT}) \leq U(T_{IN}) + U(T_{FIX}) \]
\[ \leq c_{IN} + c_{FIX} \]
How the reduction works

\[ U(\mathcal{T}_{\text{OUT}}) \leq U(\mathcal{T}_{\text{IN}}) + U(\mathcal{T}_{\text{FIX}}) \]
\[ \leq c_{\text{IN}} + c_{\text{FIX}} \]
\[ \leq c \]
## Conclusions

<table>
<thead>
<tr>
<th>Implicit deadlines $(d = p)$</th>
<th>Constrained deadlines $(d \leq p)$</th>
<th>Arbitrary deadlines $(d, p$ unrelated)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Arbitrary utilization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weakly NP-complete</td>
<td>Weakly NP-complete</td>
<td>Weakly NP-hard</td>
</tr>
<tr>
<td>Pseudo-polynomial time</td>
<td>Pseudo-polynomial time</td>
<td>Exponential time</td>
</tr>
<tr>
<td>algorithm exists</td>
<td>algorithm exists</td>
<td>algorithm exists</td>
</tr>
<tr>
<td><strong>Utilization bounded by a</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>constant $c &lt; 1$</td>
<td>Polynomial time</td>
<td></td>
</tr>
<tr>
<td>with RM priorities and</td>
<td>for $c &gt; 0$</td>
<td></td>
</tr>
<tr>
<td>$c \leq \ln(2)$</td>
<td>Pseudo-polynomial time</td>
<td></td>
</tr>
<tr>
<td>Pseudo-polynomial time</td>
<td>algorithm exists</td>
<td></td>
</tr>
<tr>
<td>algorithm exists</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Weakly NP-complete</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for $c &gt; 0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo-polynomial time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>algorithm exists</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Exponential time</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>algorithm exists</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pseudo-polynomial time</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>algorithm exists</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ponta Ekeberg**
# Conclusions

## Implicit deadlines ($d = p$)
- Weakly NP-complete
- Pseudo-polynomial time algorithm exists

## Constrained deadlines ($d \leq p$)
- Weakly NP-complete
- Pseudo-polynomial time algorithm exists

## Arbitrary deadlines ($d, p$ unrelated)
- Weakly NP-hard
- Exponential time algorithm exists

### Utilization bounded by a constant $c < 1$

<table>
<thead>
<tr>
<th>Utilization bounded by a constant $c &lt; 1$</th>
<th>Implicit deadlines ($d = p$)</th>
<th>Constrained deadlines ($d \leq p$)</th>
<th>Arbitrary deadlines ($d, p$ unrelated)</th>
</tr>
</thead>
</table>
| **Weakly NP-complete with**
  (i) RM and $c > \ln(2)$, or
  (ii) non-RM and $c > 0$ |
| Otherwise in P | Weakly NP-complete |
| Pseudo-polynomial time algorithm exists | Weakly NP-complete |
| Pseudo-polynomial time algorithm exists | Weakly NP-hard |
| for $c > 0$ | for $c > 0$ |
| Pseudo-polynomial time algorithm exists | Pseudo-polynomial time algorithm exists |

### 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

**Pseudo-polynomial time algorithm exists**

**Weakly NP-complete**

**Exponential time algorithm exists**

**Weakly NP-hard**

**Pseudo-polynomial time algorithm exists**

**Pseudo-polynomial time algorithm exists**

**Weakly NP-complete**

**Exponential time algorithm exists**

**Weakly NP-hard**

**Pseudo-polynomial time algorithm exists**

**Pseudo-polynomial time algorithm exists**
## Conclusions

<table>
<thead>
<tr>
<th>Utilization bounded by a constant $c &lt; 1$</th>
<th>Implicit deadlines ($d = p$)</th>
<th>Constrained deadlines ($d \leq p$)</th>
<th>Arbitrary deadlines ($d, p$ unrelated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary utilization</td>
<td>Weakly NP-complete</td>
<td>Weakly NP-complete</td>
<td>Weakly NP-hard</td>
</tr>
<tr>
<td></td>
<td>Pseudo-polynomial time</td>
<td>Pseudo-polynomial time</td>
<td>Exponential time</td>
</tr>
<tr>
<td></td>
<td>algorithm exists</td>
<td>algorithm exists</td>
<td>algorithm exists</td>
</tr>
<tr>
<td>Weakly NP-complete with</td>
<td>Weakly NP-complete</td>
<td>Weakly NP-complete</td>
<td></td>
</tr>
<tr>
<td>(i) RM and $c &gt; \ln(2)$, or</td>
<td>for $c &gt; 0$</td>
<td>for $c &gt; 0$</td>
<td></td>
</tr>
<tr>
<td>(ii) non-RM and $c &gt; 0$</td>
<td>Pseudo-polynomial time</td>
<td>Pseudo-polynomial time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>algorithm exists</td>
<td>algorithm exists</td>
<td></td>
</tr>
<tr>
<td>Otherwise in P</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For $c > 0$ in all cases:

- Weakly NP-hard
- Pseudo-polynomial time algorithm exists
∀Thank you!

∃Questions?