Partitioned Scheduling of Recurrent Real-Time Tasks

Pontus Ekberg
Uppsala University

Sanjoy Baruah
Washington University in Saint Louis

RTSS 2021
What is the complexity of partitioned schedulability?
What is the complexity of partitioned schedulability?

1. Exact schedulability tests!
What is the complexity of partitioned schedulability?

1. Exact schedulability tests!
2. Schedulers:
   - FP
   - EDF / feasibility
3. Tasks:
   - Synchronous periodic / sporadic
   - Asynchronous periodic
4. Processors:
   - Identical
   - Unrelated
5. ...
But, don’t we know this already?
But, don’t we know this already?

Partitioned schedulability generalizes BIN PACKING, and is therefore NP-hard!
But, don’t we know this already?

Partitioned schedulability generalizes BIN PACKING, and is therefore NP-hard!

NP-hard is just a lower bound on complexity.

What is the exact complexity?
The polynomial hierarchy

\[
\begin{align*}
\Sigma_3^P \quad & \quad \Pi_3^P \\
\Sigma_2^P \quad & \quad \Pi_2^P \\
NP \quad & \quad coNP \\
P \quad & \quad P
\end{align*}
\]
The polynomial hierarchy

\[ \vdots \]

\[ \Sigma_3^P \rightarrow \Pi_3^P \]

\[ \Sigma_2^P \rightarrow \Pi_2^P \]

NP \rightarrow coNP

\[ \vdots \]

\[ P \]
The polynomial hierarchy

\[ \vdots \]

\[ \Sigma^p_3 \quad \Pi^p_3 \]

\[ \Sigma^p_2 \quad \Pi^p_2 \]

\[ \text{NP} \quad \text{coNP} \]

\[ \text{P} \]
THE POLYNOMIAL HIERARCHY

\[
\begin{aligned}
& \Sigma^P_3 & \Pi^P_3 \\
& \Sigma^P_2 & \Pi^P_2 \\
& \text{NP} & \text{coNP} \\
& \text{P} & \text{P} \\
\end{aligned}
\]
The polynomial hierarchy

\[ \text{NP}^\text{NP} = \Sigma_2^\text{P} = \Pi_2^\text{P} \]

\[ \text{NP} \quad \text{coNP} \]

\[ \text{P} \]
The polynomial hierarchy

\[ \vdots \]

\[ \Sigma_3^P \quad \Pi_3^P \]

\[ \cdots = \Sigma_2^P \quad \Pi_2^P = \text{coNP}^{\text{NP}} \]

\[ \text{NP}^{\text{NP}} = \Sigma_2^P \quad \Pi_2^P = \text{coNP}^{\text{NP}} \]

\[ \text{NP} \quad \text{coNP} \]

\[ \vdots \quad \vdots \]

\[ \text{P} \]
The polynomial hierarchy

\[
\begin{align*}
\text{NP}^{\Sigma_2^P} &= \Sigma_3^P \\
\text{NP}^{NP} &= \Sigma_2^P \\
\text{NP} &= \text{coNP} \\
P &= P
\end{align*}
\]
The polynomial hierarchy

\[ \begin{align*}
NP^{\Sigma_2^P} &= \Sigma_3^P \\
NP^{NP} &= \Sigma_2^P \\
NP &= \Sigma_1^P \\
coNP &= \Sigma_1^C \\
P &= \Sigma_0^P \\
\Pi_2^P &= \text{coNP}^{NP} \\
\Pi_3^P &= \text{coNP}^{\Sigma_2^P}
\end{align*} \]
The polynomial hierarchy

\[ \vdots \quad \vdots \]

\[ \Sigma_3^P \quad \Pi_3^P \]

\[ \Sigma_2^P \quad \Pi_2^P \]

\[ \text{NP} \quad \text{coNP} \]

\[ \text{P} \]

\[ \varphi(x) \]
The polynomial hierarchy

\[
\vdots
\]

\[
\Sigma_3^P \quad \Pi_3^P
\]

\[
\Sigma_2^P \quad \Pi_2^P
\]

\[
\exists w. \varphi(x, w)
\]

NP \quad coNP

\[
\varphi(x)
\]

\[
P
\]
The polynomial hierarchy
The polynomial hierarchy

\[ \begin{align*}
\exists w_1 \forall w_2. \varphi(x, w_1, w_2) \\
\exists w. \varphi(x, w) \\
\forall w. \varphi(x, w) \\
\varphi(x)
\end{align*} \]
The polynomial hierarchy

\[ \Sigma_2^P \quad \Pi_2^P \]

\[ \Sigma_3^P \quad \Pi_3^P \]

\[ \exists w_1 \forall w_2. \varphi(x, w_1, w_2) \quad \forall w_1 \exists w_2. \varphi(x, w_1, w_2) \]

\[ \exists w. \varphi(x, w) \quad \forall w. \varphi(x, w) \]

\[ \exists w_1 \forall w_2 \exists w_3. \varphi(x, w_1, w_2, w_3) \quad \forall w_1 \exists w_2 \forall w_3. \varphi(x, w_1, w_2, w_3) \]

\[ \varphi(x) \]

\[ \exists w_1 \forall w_2. \varphi(x, w_1, w_2) \quad \forall w_1 \exists w_2. \varphi(x, w_1, w_2) \]

\[ \exists w. \varphi(x, w) \quad \forall w. \varphi(x, w) \]

\[ \exists w_1 \forall w_2 \exists w_3. \varphi(x, w_1, w_2, w_3) \quad \forall w_1 \exists w_2 \forall w_3. \varphi(x, w_1, w_2, w_3) \]

\[ \varphi(x) \]
The polynomial hierarchy

\[ \exists w_1 \forall w_2 \exists w_3. \varphi(x, w_1, w_2, w_3) \]

\[ \exists w_1 \forall w_2. \varphi(x, w_1, w_2) \]

\[ \forall w_1 \exists w_2. \varphi(x, w_1, w_2) \]

\[ \exists w. \varphi(x, w) \]

\[ \forall w. \varphi(x, w) \]

\[ \varphi(x) \]
The polynomial hierarchy

Partitioned schedulability problems

\[ \Sigma_2^P \quad \Pi_2^P \]

\[ \Sigma_3^P \quad \Pi_3^P \]

\[ \text{NP} \quad \text{coNP} \]

\[ \text{P} \]
The polynomial hierarchy

Partitioned schedulability problems

Problems that can be efficiently formulated as ILP or SAT
The polynomial hierarchy

Partitioned schedulability problems

Problems that can be efficiently formulated as ILP or SAT
Complexity for sporadic / synchronous periodic tasks
Complexity for sporadic / synchronous periodic tasks

\[
\begin{array}{c}
\vdots \\
\Sigma_3^P & \Pi_3^P \\
\Sigma_2^P & \Pi_2^P \\
NP & \text{coNP} \\
P \\
\end{array}
\]
Complexity for sporadic / synchronous periodic tasks

- EDF with implicit deadlines
- FP with arbitrary deadlines
- Problems that can be efficiently formulated as ILP or SAT
- EDF with implicit deadlines
- FP with implicit or constrained deadlines
- ILPs in paper
- $P \subseteq \Sigma_1^P \subseteq \Pi_2^P \subseteq \Sigma_2^P \subseteq \Pi_3^P \subseteq \Sigma_3^P \subseteq \Pi_3^P \subseteq \Sigma_2^P \subseteq \Pi_2^P \subseteq \Sigma_1^P \subseteq P \subseteq \text{coNP}$
Complexity for sporadic / synchronous periodic tasks

- EDF with constrained or arbitrary deadlines
- EDF with implicit deadlines

- NP
- coNP
- P

Classifications:
- \( \Sigma_2^P \)
- \( \Pi_2^P \)
- \( \Sigma_3^P \)
- \( \Pi_3^P \)

Complex classes:
- \( \Sigma_3 \)
- \( \Pi_3 \)
- \( \Sigma_2 \)
- \( \Pi_2 \)
Complexity for sporadic / synchronous periodic tasks

- EDF with constrained or arbitrary deadlines
- EDF with implicit deadlines
- FP with implicit or constrained deadlines

Complexity Classes:
- \( \Sigma_2^P \)
- \( \Sigma_3^P \)
- \( \Pi_2^P \)
- \( \Pi_3^P \)
- NP
- coNP
- P

- NP
- coNP
- P

5
Complexity for sporadic / synchronous periodic tasks

- EDF with constrained or arbitrary deadlines
- EDF with implicit deadlines
- FP with implicit or constrained deadlines
- FP with arbitrary deadlines

NP

P

coNP

ΣP2

ΠP2

ΣP3

ΠP3

...
Complexity for sporadic / synchronous periodic tasks

- Problems that can be efficiently formulated as ILP or SAT
- EDF with constrained or arbitrary deadlines
- EDF with implicit deadlines
- FP with implicit or constrained deadlines
- FP with arbitrary deadlines

- NP
- coNP
- P
- \( \Sigma_p^P \)
- \( \Pi_p^P \)
- \( \Sigma_3^P \)
- \( \Pi_3^P \)
Complexity for sporadic / synchronous periodic tasks

- Problems that can be efficiently formulated as ILP or SAT
- FP with implicit or constrained deadlines
- EDF with implicit deadlines
- EDF with constrained or arbitrary deadlines
- ILPs in paper

Complexity classes:
- \( \Sigma_2 \)
- \( \Pi_2 \)
- \( \Sigma_3 \)
- \( \Pi_3 \)
Simultaneous Congruences

The Simultaneous Congruences Problem (scp):

Example:

\[ A = \{(2, 4), (4, 6), (3, 8), (0, 3)\} \]

\[ k = 2 \]

scp is \( \text{NP} \)-complete (Leung and Whitehead, 1982)
Simultaneous Congruences

The Simultaneous Congruences Problem (scp):

Example: \( A = \{(2, 4), (4, 6), (3, 8), (0, 3)\}, \quad k = 2 \)
The Simultaneous Congruences Problem (scp):

Example: \[ A = \{(2, 4), (4, 6), (3, 8), (0, 3)\}, \quad k = 2 \]
Simultaneous Congruences

The Simultaneous Congruences Problem (scp):

Example: \[ A = \{(2, 4), (4, 6), (3, 8), (0, 3)\}, \quad k = 2 \]
The Simultaneous Congruences Problem (scp):

Example: \[ A = \{(2, 4), (4, 6), (3, 8), (0, 3)\}, \quad k = 2 \]

scp is NP-complete (Leung and Whitehead, 1982)
Let’s generalize it!
Let’s generalize it!

**Example:** \[ A = \{(2, 4), (4, 6), (3, 8), (0, 3), \ldots\} \]

**Partitioned SCP:**

\[
\sum_{P} P_{2} \text{-complete, Rutenburg, 1986}
\]

\[
\sum_{P} P_{2} \text{-hard}
\]
Let’s generalize it!

**PARTITIONED SCP:**

Example: \( A = \{(2, 4), (4, 6), (3, 8), (0, 3), \ldots\} \), \( m, k \)

\[ A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_m \]
Let’s generalize it!

**Example:** \( A = \{(2, 4), (4, 6), (3, 8), (0, 3), \ldots \} \), \( m, k \)

**Partitioned SCP:**

\[
A_1 \quad A_2 \quad \cdots \quad A_m
\]

\[
(A_1, k) \not\in \text{SCP} \quad (A_2, k) \not\in \text{SCP} \quad \cdots \quad (A_m, k) \not\in \text{SCP}
\]

**Greedily colored graph partitioned SCP:**

Some partitioned schedulability problems \( \Sigma P_2 \)-complete, Rutenburg, 1986

\( \Sigma P_2 \)-complete

\( \Sigma P_2 \)-hard
Let’s generalize it!

**Partitioned SCP:**

Example: \( A = \{ (2, 4), (4, 6), (3, 8), (0, 3), \ldots \} \), \( m \), \( k \)

\[
\begin{align*}
A_1 & \quad A_2 \quad \cdots \quad A_m \\
(A_1, k) & \not\in \text{SCP} \quad (A_2, k) & \not\in \text{SCP} \quad \cdots \quad (A_m, k) & \not\in \text{SCP}
\end{align*}
\]

**Generalized Graph Coloring** \(\rightarrow\) **Partitioned SCP**
Let’s generalize it!

Example: \( A = \{(2, 4), (4, 6), (3, 8), (0, 3), \ldots\}, \ m, \ k \)

\[ A_1 \leftarrow A_2 \cdots \rightarrow A_m \]

\[ (A_1, k) \not\in \text{SCP} \]
\[ (A_2, k) \not\in \text{SCP} \]
\[ \cdots \]
\[ (A_m, k) \not\in \text{SCP} \]

Generalized Graph Coloring \( \rightarrow \) Partitioned SCP \( \rightarrow \) Some partitioned schedulability problems
Let’s generalize it!

**PARTITIONED SCP:**

Example: \( A = \{(2, 4), (4, 6), (3, 8), (0, 3), \ldots\} \) \( m, k \)

\[
\begin{align*}
A_1 & \quad \quad A_2 & \quad \quad \cdots & \quad \quad A_m \\
(A_1, k) & \not\in \text{SCP} & (A_2, k) & \not\in \text{SCP} & \cdots & \not\in \text{SCP} & (A_m, k) & \not\in \text{SCP}
\end{align*}
\]

**GENERALIZED GRAPH COLORING** → **PARTITIONED SCP** → Some partitioned schedulability problems

\( \Sigma^P_2 \)-complete, Rutenburg, 1986
Let’s generalize it!

**Partitioned SCP:**

Example: \( A = \{(2, 4), (4, 6), (3, 8), (0, 3), \ldots\}, \ m, \ k \)

\[ A_1 \quad A_2 \quad \ldots \quad A_m \]

\( (A_1, k) \notin \text{SCP} \quad (A_2, k) \notin \text{SCP} \quad \ldots \quad (A_m, k) \notin \text{SCP} \)

**Generalized Graph Coloring** \( \rightarrow \) **Partitioned SCP** \( \rightarrow \) Some partitioned schedulability problems

\( \Sigma_2^P \)-complete, Rutenburg, 1986

\( \Sigma_2^P \)-complete
Let’s generalize it!

Partitioned SCP:

Example: \( A = \{(2, 4), (4, 6), (3, 8), (0, 3), \ldots\}, \ m, \ k \)

\[
A_1 \quad \quad A_2 \quad \quad \cdots \quad \quad A_m
\]

\((A_1, k) \notin \text{SCP}\)
\((A_2, k) \notin \text{SCP}\)
\((A_m, k) \notin \text{SCP}\)

Generalized Graph Coloring \(\Sigma_p^P\)-complete, Rutenburg, 1986

Some partitioned schedulability problems

\(\Sigma_p^P\)-complete

\(\Sigma_2^P\)-hard
Complexity for Asynchronous Periodic Tasks
Complexity for asynchronous periodic tasks

Any work-conserving scheduler with constrained deadlines can formulate problems efficiently as ILP or SAT. EDF with implicit deadlines or EDF with constrained or arbitrary deadlines.

\[
\begin{array}{ccc}
\Sigma_3^P & \quad & \Pi_3^P \\
\Sigma_2^P & \quad & \Pi_2^P \\
NP & \quad & coNP \\
P & \quad & \\
\end{array}
\]
Complexity for asynchronous periodic tasks

Any work-conserving scheduler with constrained deadlines can efficiently formulate problems as ILP or SAT. EDF with implicit deadlines is in P, and EDF with constrained or arbitrary deadlines is in coNP.
Complexity for asynchronous periodic tasks

Any work-conserving scheduler with constrained deadlines can be formulated as an FPs problem, which can be efficiently formulated as an ILP or SAT problem. EDF with implicit deadlines is in NP, and EDF with constrained or arbitrary deadlines is in \( \Sigma_2^P \). EDF with implicit deadlines is in \( \Pi_2^P \).
Complexity for asynchronous periodic tasks

Any work-conserving scheduler with constrained deadlines.

Problems that can be efficiently formulated as ILP or SAT.

EDF with constrained or arbitrary deadlines.

NP

EDF with implicit deadlines.

Constrained or arbitrary deadlines.
Complexity for asynchronous periodic tasks

Any work-conserving scheduler with constrained deadlines

FP

Problems that can be efficiently formulated as ILP or SAT

EDF with constrained or arbitrary deadlines

P

NP

coNP

Any work-conserving scheduler with constrained deadlines

EDF with implicit deadlines

Σ₂

Π₂

Σ₃

Π₃
Complexity for asynchronous periodic tasks

Any work-conserving scheduler with constrained deadlines

Problems that can be efficiently formulated as ILP or SAT

EDF with constrained or arbitrary deadlines

EDF with implicit deadlines

Any work-conserving scheduler with constrained deadlines

Problems that can be efficiently formulated as ILP or SAT
Conclusions

New complexity bounds for partitioned schedulability.
Conclusions

New complexity bounds for partitioned schedulability.

- Some problems are exactly pinpointed.
- Some are provably beyond the corresponding uniprocessor case.
- Some are essentially the same as the uniprocessor case!
- Some can not be formulated as ILP in polynomial time.
- No problem is higher up than $\Sigma^P_3$.

†: Unless the polynomial hierarchy collapses
Conclusions

New complexity bounds for partitioned schedulability.

- Some problems are exactly pinpointed.
- Some are provably† beyond the corresponding uniprocessor case.

†: Unless the polynomial hierarchy collapses
Conclusions

New complexity bounds for partitioned schedulability.

• Some problems are exactly pinpointed.
• Some are provably† beyond the corresponding uniprocessor case.
• Some are essentially the same as the uniprocessor case!

†: Unless the polynomial hierarchy collapses
Conclusions

New complexity bounds for partitioned schedulability.

- Some problems are exactly pinpointed.
- Some are provably† beyond the corresponding uniprocessor case.
- Some are essentially the same as the uniprocessor case!
- Some can not be formulated as ILP in polynomial time.

†: Unless the polynomial hierarchy collapses
Conclusions

New complexity bounds for partitioned schedulability.

- Some problems are exactly pinpointed.
- Some are provably† beyond the corresponding uniprocessor case.
- Some are essentially the same as the uniprocessor case!
- Some can not be formulated as ILP in polynomial time.
- No problem is higher up than \( \Sigma_3^P \).

†: Unless the polynomial hierarchy collapses
∀Thank you!

∃Questions?