Complexity of partitioned scheduling for periodic tasks

Pontus Ekberg
Uppsala University

Sanjoy Baruah
Washington University in Saint Louis

MAPSP'22

Periodic tasks

A periodic task τ_i is given by $(O_i, C_i, D_i, T_i) \in \mathbb{N}^4$, where

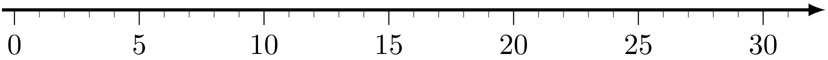
- O_i is the initial offset,
- C_i is the worst-case execution time,
- D_i is the relative deadline, and
- T_i is the period.

Periodic tasks

A periodic task τ_i is given by $(O_i, C_i, D_i, T_i) \in \mathbb{N}^4$, where

- O_i is the initial offset,
- C_i is the worst-case execution time,
- D_i is the relative deadline, and
- T_i is the period.

$$\tau_i = (O_i, C_i, D_i, T_i) = (4, 3, 6, 10)$$

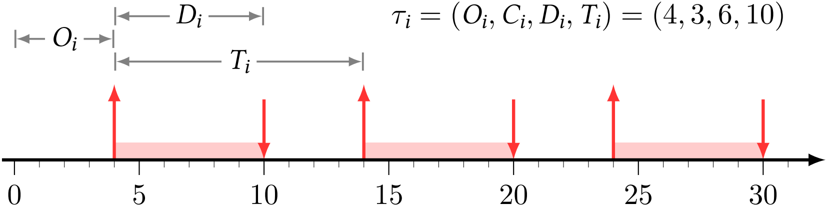


Periodic tasks

A periodic task τ_i is given by $(O_i, C_i, D_i, T_i) \in \mathbb{N}^4$, where

- O_i is the initial offset,
- C_i is the worst-case execution time,
- D_i is the relative deadline, and
- T_i is the period.

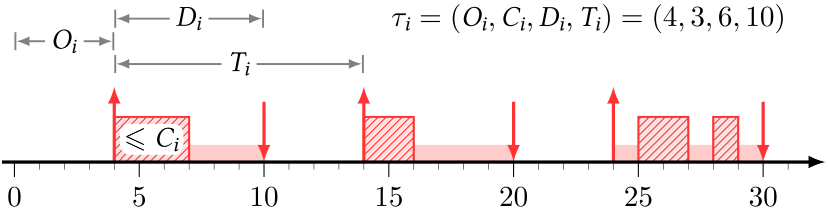
$$\tau_i = (O_i, C_i, D_i, T_i) = (4, 3, 6, 10)$$



Periodic tasks

A periodic task τ_i is given by $(O_i, C_i, D_i, T_i) \in \mathbb{N}^4$, where

- O_i is the initial offset,
- C_i is the worst-case execution time,
- D_i is the relative deadline, and
- T_i is the period.



Common restrictions

Let $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ be a set of tasks.

We say that $\mathcal{T}$ has

- *implicit deadlines*, if $D_i = T_i$ for all $\tau_i \in \mathcal{T}$,
- *constrained deadlines*, if $D_i \leqslant T_i$ for all $\tau_i \in \mathcal{T}$,
- *arbitrary deadlines*, otherwise.

Also, $\mathcal{T}$ is

- *synchronous*, if $O_i = O_j$ for all $\tau_i, \tau_j \in \mathcal{T}$,
- *asynchronous*, otherwise.

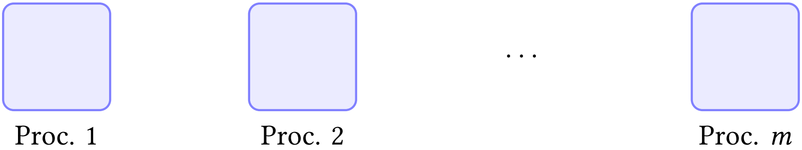
Partitioned scheduling

Partitioned scheduling

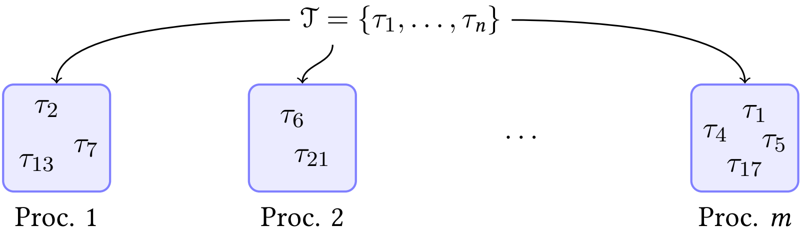
$$\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$$

Partitioned scheduling

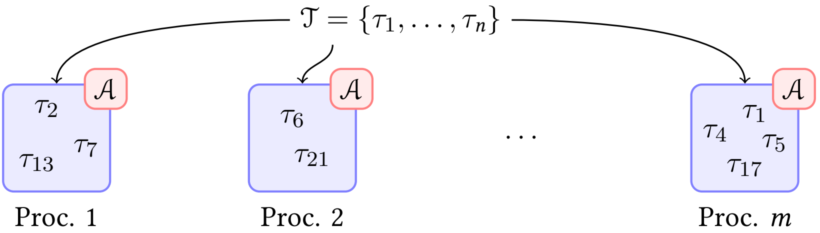
$$\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$$



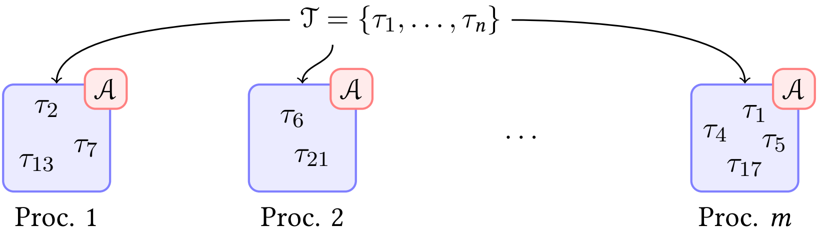
Partitioned scheduling



Partitioned scheduling



Partitioned scheduling



PARTITIONED $\mathcal{A}$-SCHEDULABILITY

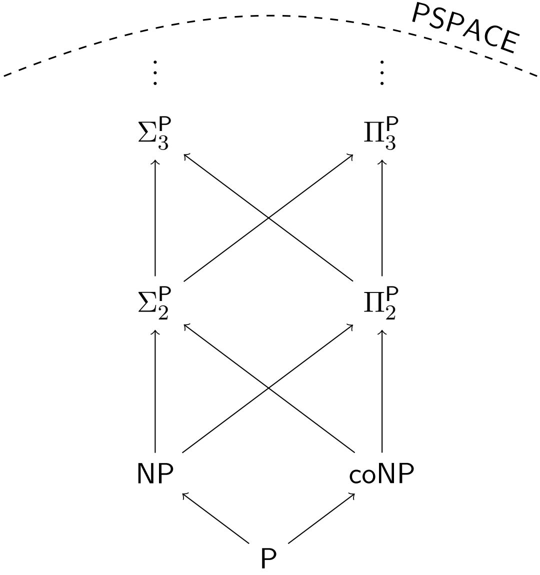
Instance: $\langle \mathcal{T}, m \rangle$, where $\mathcal{T}$ is a set of tasks and m is the number of processors.

Question: Is there a partitioning $(\mathcal{T}_1, \ldots, \mathcal{T}_m)$ of $\mathcal{T}$ such that each $\mathcal{T}_i$ is schedulable by $\mathcal{A}$ on a single processor?

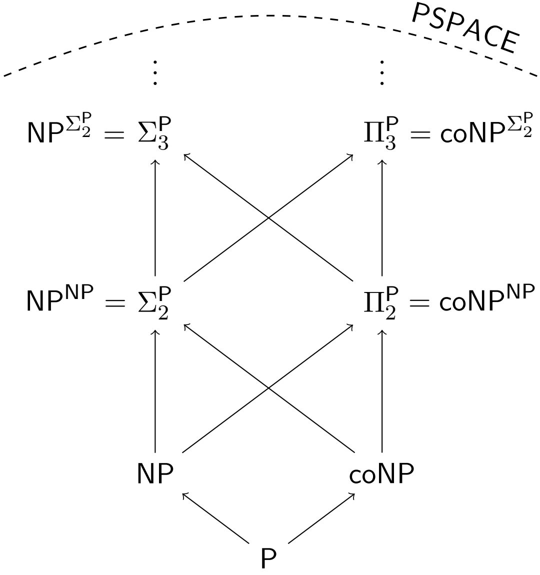
The polynomial hierarchy

$$
\begin{tikzcd}
\vdots & & \vdots \\
\Sigma_3^{\mathsf{P}} & & \Pi_3^{\mathsf{P}} \\
\Sigma_2^{\mathsf{P}} \arrow[u] \arrow[urr] & & \Pi_2^{\mathsf{P}} \arrow[u] \arrow[ull] \\
\mathsf{NP} \arrow[u] \arrow[urr] & & \mathsf{coNP} \arrow[u] \arrow[ull] \\
& \mathsf{P} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

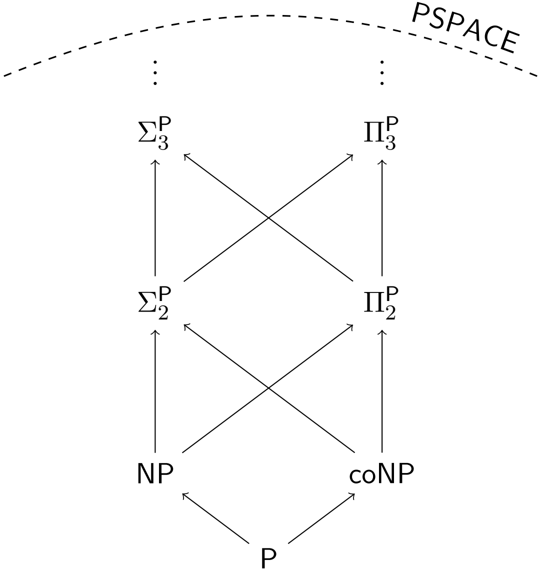
The polynomial hierarchy



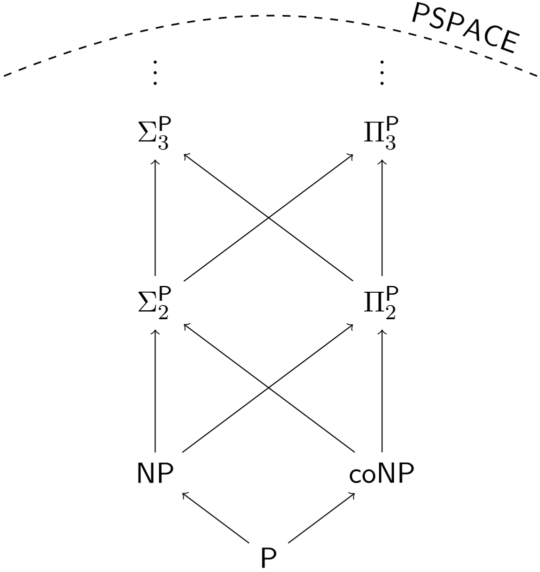
The polynomial hierarchy



The polynomial hierarchy

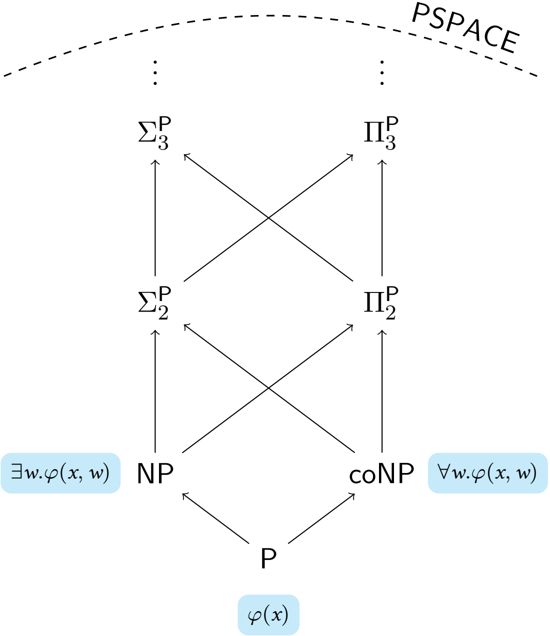


The polynomial hierarchy

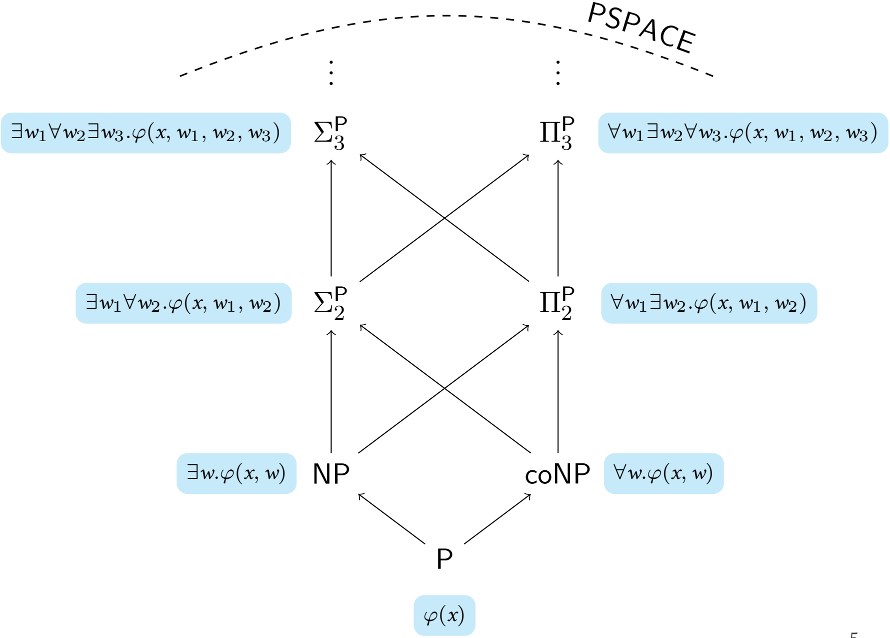


$\varphi(x)$

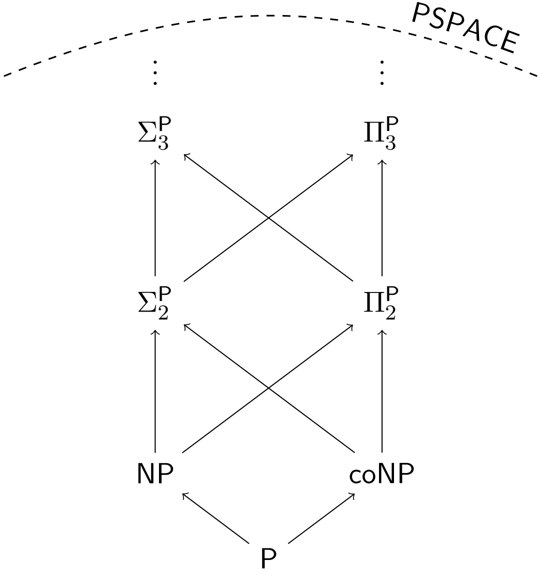
The polynomial hierarchy



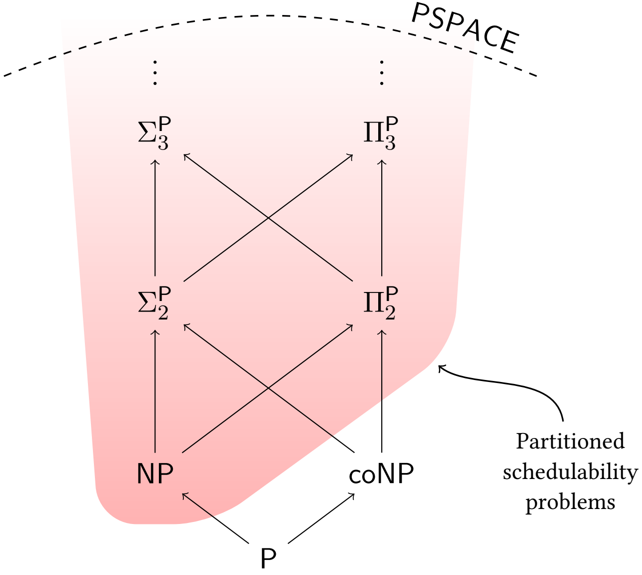
The polynomial hierarchy



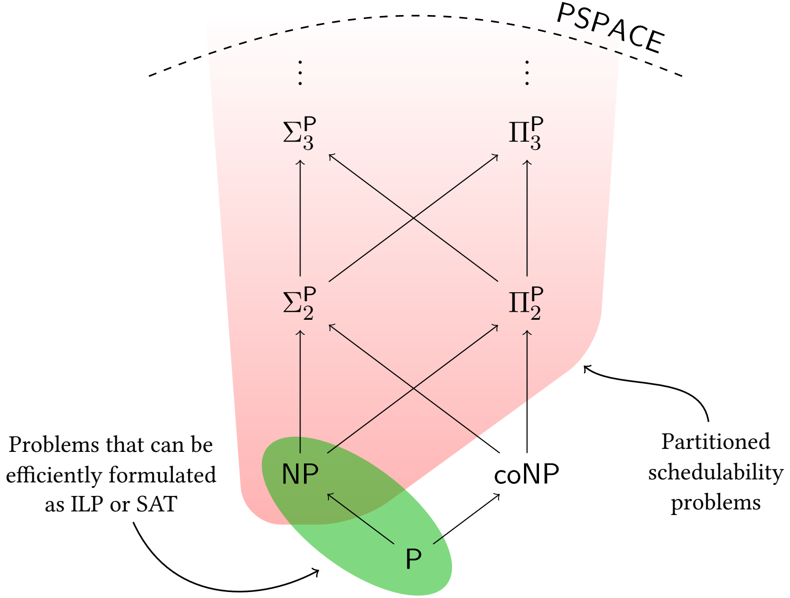
The polynomial hierarchy



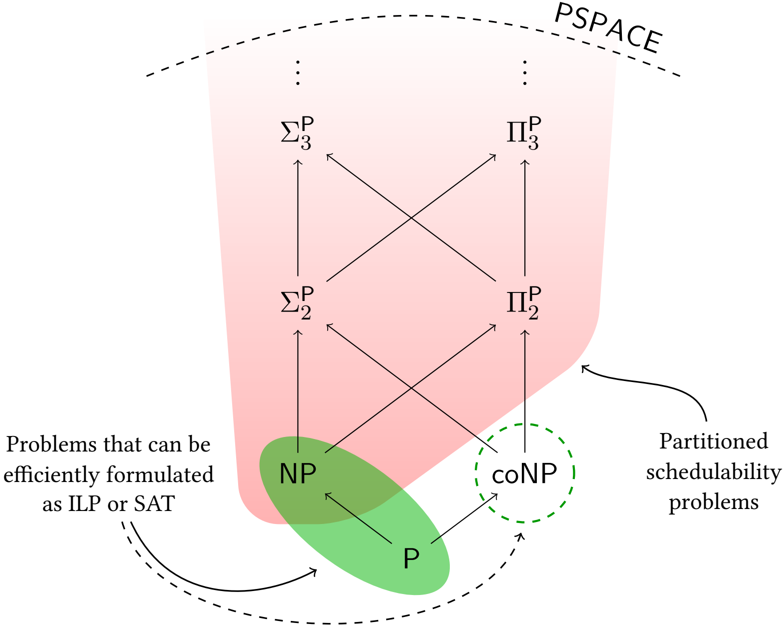
The polynomial hierarchy



The polynomial hierarchy



The polynomial hierarchy



Simultaneous Congruences

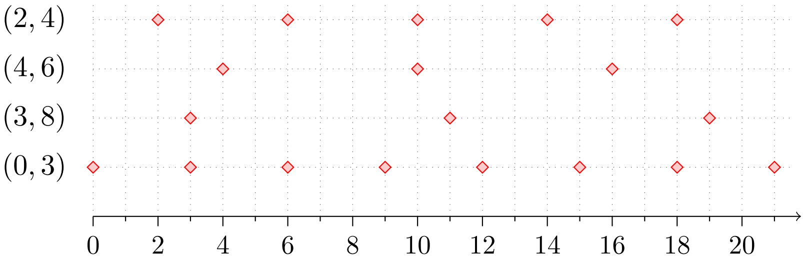
The Simultaneous Congruences Problem (SCP):

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

Simultaneous Congruences

The Simultaneous Congruences Problem (SCP):

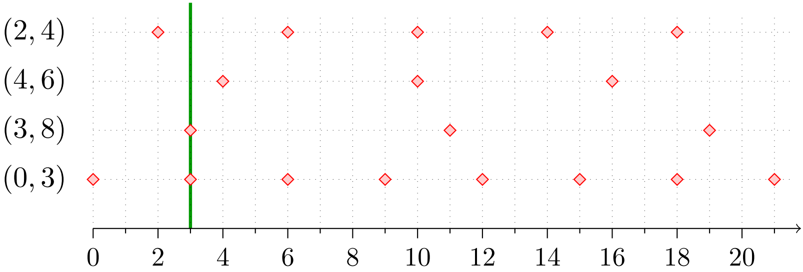
Example: $A = \{(2,4),(4,6),(3,8),(0,3)\}, \qquad k = 2$



Simultaneous Congruences

The Simultaneous Congruences Problem (SCP):

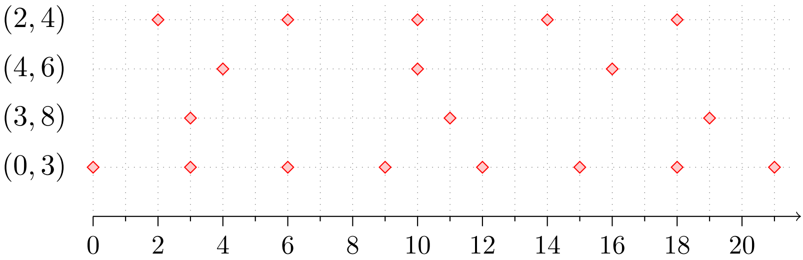
Example: $A = \{(2,4), (4,6), (3,8), (0,3)\}, \qquad k = 2$



Simultaneous Congruences

The Simultaneous Congruences Problem (SCP):

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

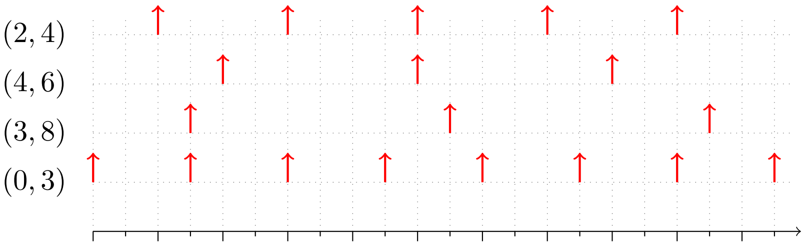


SCP is NP-complete (Leung and Whitehead, 1982)

Simultaneous Congruences

The Simultaneous Congruences Problem (SCP):

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

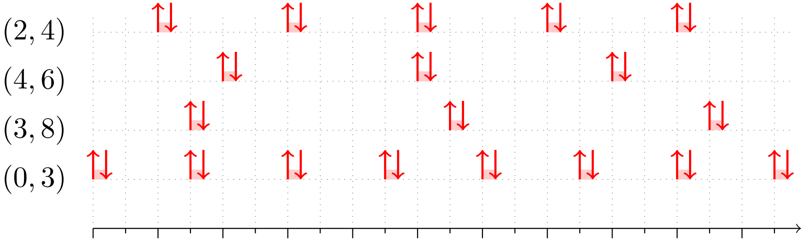


SCP is NP-complete (Leung and Whitehead, 1982)

Simultaneous Congruences

The Simultaneous Congruences Problem (SCP):

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



SCP is NP-complete (Leung and Whitehead, 1982)

Let's generalize scp!

Let's generalize SCP!

PARTITIONED SCP

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

Let's generalize SCP!

PARTITIONED SCP

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

$$A_1 \qquad A_2 \qquad \cdots \qquad A_m$$

Let's generalize SCP!

PARTITIONED SCP

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

$$
\begin{tikzcd}
 & & A = \{(2,4),(4,6),(3,8),(0,3),\ldots\} \arrow[dll] \arrow[dl] \arrow[drr] & & \\
A_1 \arrow[d, dash] & A_2 \arrow[d, dash] & \cdots & & A_m \arrow[d, dash] \\
\langle A_1, k\rangle \notin \textsc{scp} & \langle A_2, k\rangle \notin \textsc{scp} & \cdots & & \langle A_m, k\rangle \notin \textsc{scp}
\end{tikzcd}
$$

Let's generalize scp!

partitioned scp

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

$$\begin{array}{cccc} A_1 & A_2 & \cdots & A_m \\ \langle A_1, k\rangle \notin \textsc{scp} & \langle A_2, k\rangle \notin \textsc{scp} & \cdots & \langle A_m, k\rangle \notin \textsc{scp} \end{array}$$

Let's generalize SCP!

PARTITIONED SCP

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

$$\begin{array}{cccc} A_1 & A_2 & \cdots & A_m \\ | & | & & | \\ \langle A_1, k\rangle \notin \text{SCP} & \langle A_2, k\rangle \notin \text{SCP} & \cdots & \langle A_m, k\rangle \notin \text{SCP} \end{array}$$

GENERALIZED GRAPH COLORING $\longrightarrow$ PARTITIONED SCP

Let's generalize SCP!

PARTITIONED SCP

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

$$A_1 \qquad A_2 \qquad \cdots \qquad A_m$$

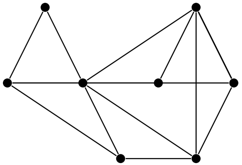
$$\langle A_1, k\rangle \notin \textsc{scp} \qquad \langle A_2, k\rangle \notin \textsc{scp} \qquad \cdots \qquad \langle A_m, k\rangle \notin \textsc{scp}$$

GENERALIZED GRAPH COLORING $\longrightarrow$ PARTITIONED SCP $\longrightarrow$ Some partitioned schedulability problems

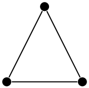
Hardness of partitioned scp

GENERALIZED GRAPH COLORING

G

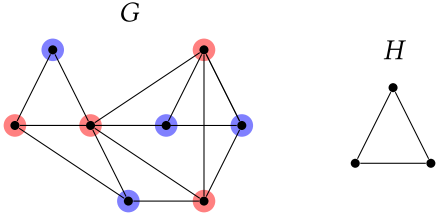


H



Hardness of partitioned scp

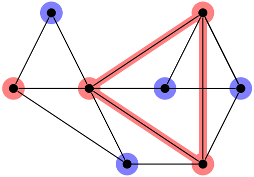
GENERALIZED GRAPH COLORING



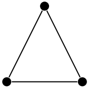
Hardness of partitioned scp

GENERALIZED GRAPH COLORING

G

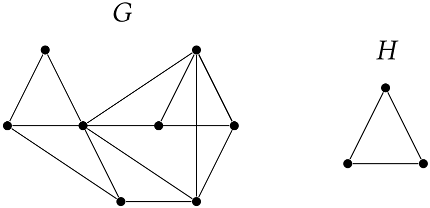


H



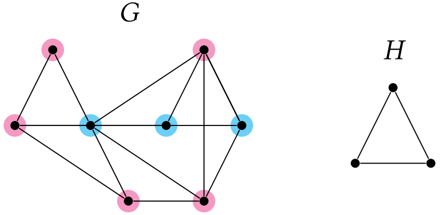
Hardness of partitioned SCP

GENERALIZED GRAPH COLORING



Hardness of partitioned scp

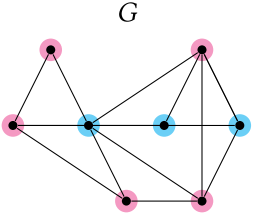
GENERALIZED GRAPH COLORING

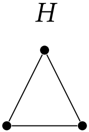


Hardness of partitioned SCP

GENERALIZED GRAPH COLORING

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)



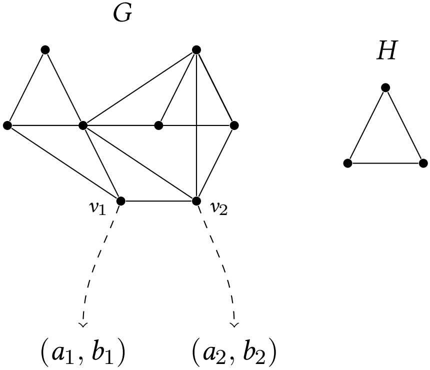


Hardness of partitioned SCP

GENERALIZED GRAPH COLORING

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)

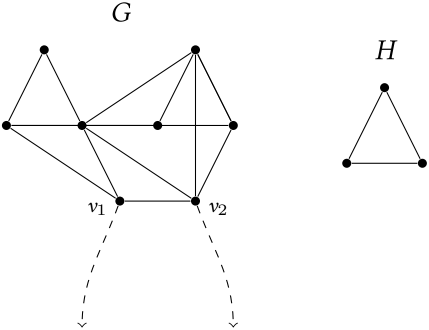
PARTITIONED SCP



Hardness of partitioned SCP

GENERALIZED GRAPH COLORING

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)



PARTITIONED SCP

(a_1, b_1) (a_2, b_2)

(a_7, b_7) (a_4, b_4)

(a_5, b_5)

(a_6, b_6)

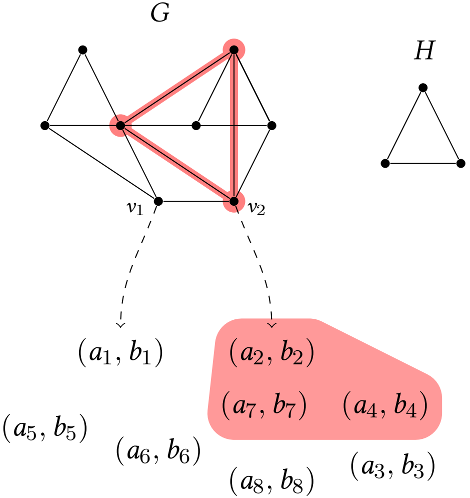
(a_8, b_8) (a_3, b_3)

Hardness of partitioned SCP

GENERALIZED GRAPH COLORING

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)

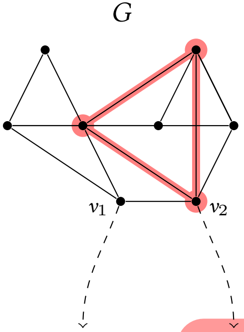
PARTITIONED SCP

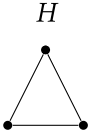


Hardness of partitioned SCP

GENERALIZED GRAPH COLORING

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)





This method from Leung and Whitehead, 1982

PARTITIONED SCP

(a_1, b_1) (a_2, b_2)

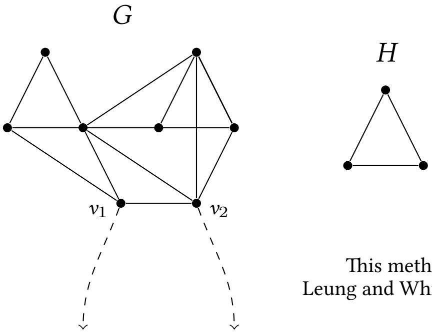
(a_7, b_7) (a_4, b_4)

(a_5, b_5) (a_6, b_6) (a_8, b_8) (a_3, b_3)

Hardness of partitioned SCP

Generalized Graph Coloring

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)



This method from Leung and Whitehead, 1982

Partitioned SCP

(a_1, b_1) (a_2, b_2)

(a_7, b_7) (a_4, b_4)

(a_5, b_5)

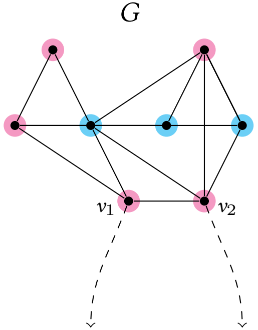
(a_6, b_6)

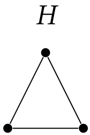
(a_8, b_8) (a_3, b_3)

Hardness of partitioned SCP

GENERALIZED GRAPH COLORING

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)





This method from Leung and Whitehead, 1982

PARTITIONED SCP

(a_1, b_1) (a_2, b_2)

(a_7, b_7) (a_4, b_4)

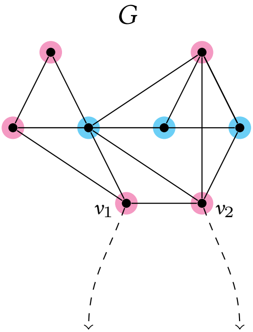
(a_5, b_5) (a_6, b_6)

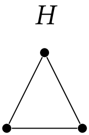
(a_8, b_8) (a_3, b_3)

Hardness of partitioned SCP

Generalized Graph Coloring

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)





This method from Leung and Whitehead, 1982

Partitioned SCP

(a_1, b_1) (a_2, b_2)

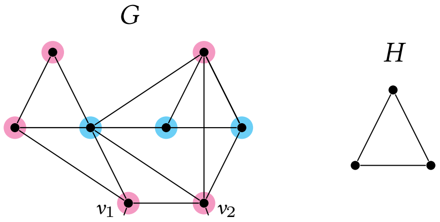
(a_7, b_7) (a_4, b_4)

(a_5, b_5) (a_6, b_6) (a_8, b_8) (a_3, b_3)

#colliding congruences $(k) = |H|$

#partitions $(m) =$ #colors

Hardness of partitioned SCP



Generalized Graph Coloring

Σ_2^{P}-complete with two colors and H is a complete graph (Rutenburg, 1986)

This method from Leung and Whitehead, 1982

Partitioned SCP

Σ_2^{P}-complete

(a_1, b_1) (a_2, b_2)

(a_7, b_7) (a_4, b_4)

(a_5, b_5) (a_6, b_6)

(a_8, b_8) (a_3, b_3)

#colliding congruences $(k) = |H|$

#partitions $(m) =$ #colors

Let's generalize SCP!

PARTITIONED SCP

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

$$\begin{array}{cccc} A_1 & A_2 & \cdots & A_m \\ \langle A_1, k\rangle \notin \textsc{scp} & \langle A_2, k\rangle \notin \textsc{scp} & \cdots & \langle A_m, k\rangle \notin \textsc{scp} \end{array}$$

GENERALIZED GRAPH COLORING → PARTITIONED SCP → Some partitioned schedulability problems

Let's generalize SCP!

PARTITIONED SCP

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

$$A_1 \qquad A_2 \qquad \cdots \qquad A_m$$

$$\langle A_1, k\rangle \notin \text{SCP} \qquad \langle A_2, k\rangle \notin \text{SCP} \qquad \cdots \qquad \langle A_m, k\rangle \notin \text{SCP}$$

GENERALIZED GRAPH COLORING $\longrightarrow$ PARTITIONED SCP $\longrightarrow$ Some partitioned schedulability problems

Σ_2^{P}-complete, Rutenburg, 1986

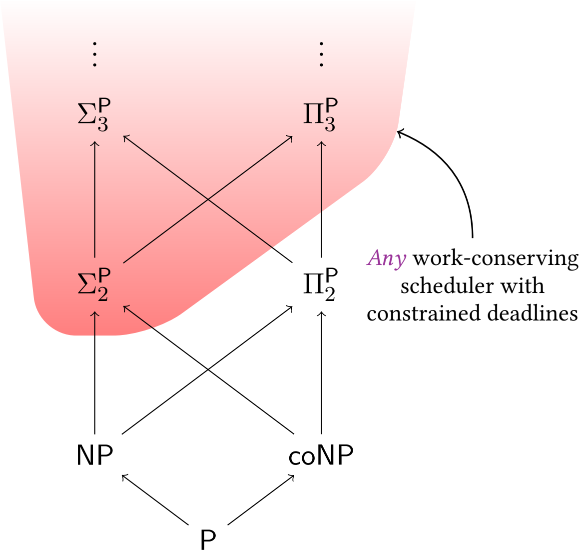
Σ_2^{P}-complete

Σ_2^{P}-hard

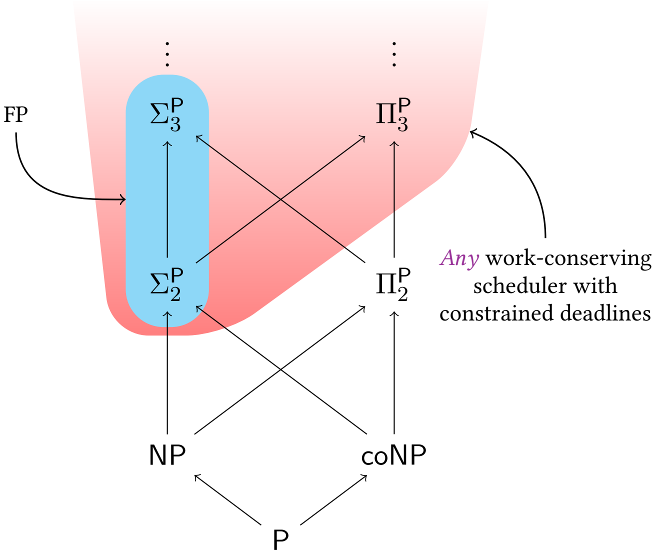
Complexity for asynchronous periodic tasks

$$
\begin{tikzcd}
\vdots & & \vdots \\
\Sigma_3^{\mathsf{P}} & & \Pi_3^{\mathsf{P}} \\
\Sigma_2^{\mathsf{P}} \arrow[u] \arrow[urr] & & \Pi_2^{\mathsf{P}} \arrow[u] \arrow[ull] \\
\mathsf{NP} \arrow[u] \arrow[urr] & & \mathsf{coNP} \arrow[u] \arrow[ull] \\
& \mathsf{P} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

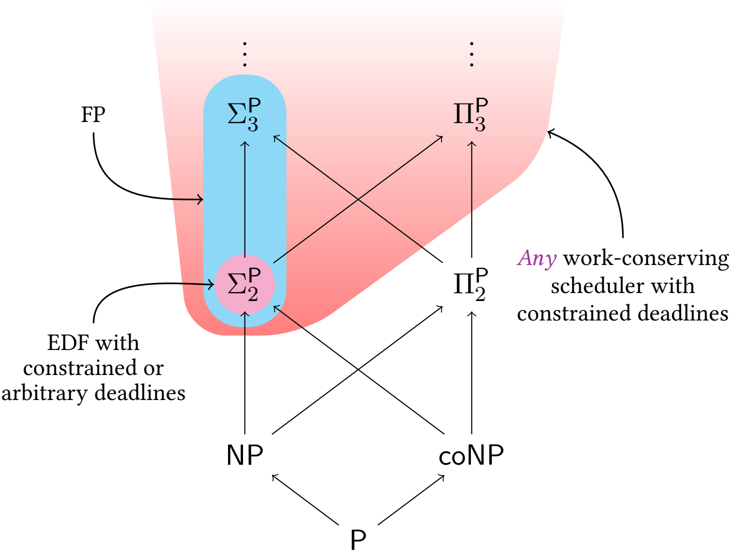
Complexity for asynchronous periodic tasks



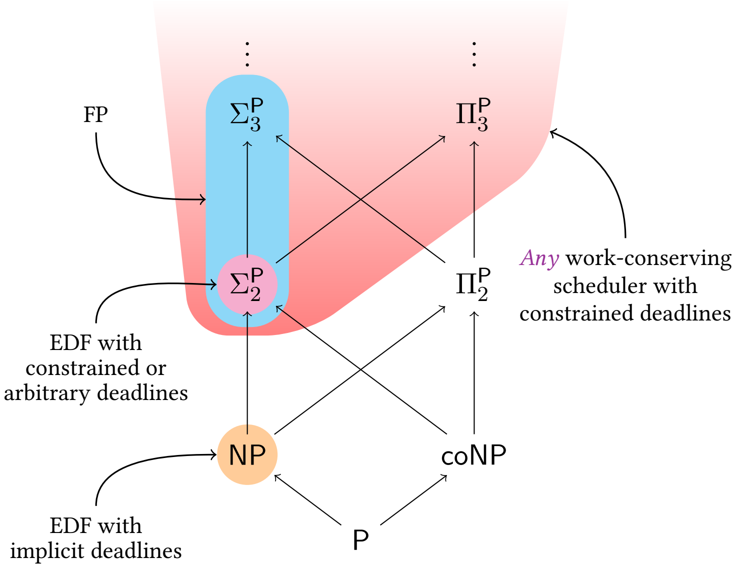
Complexity for asynchronous periodic tasks



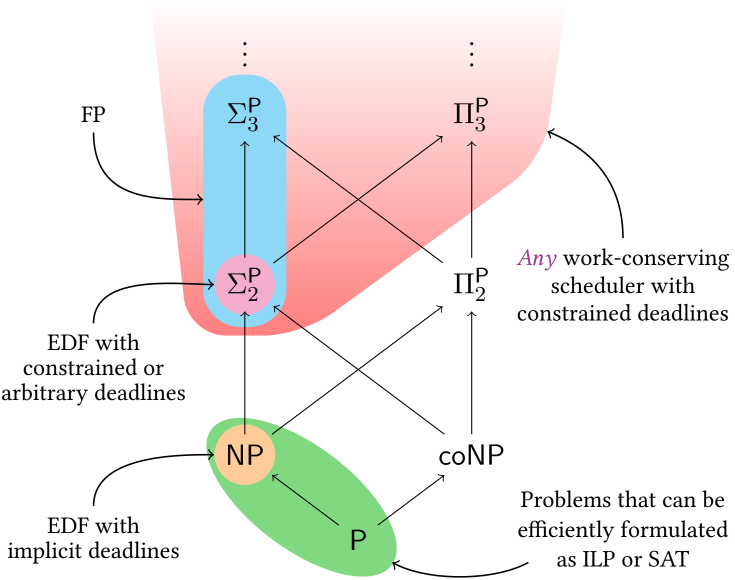
Complexity for asynchronous periodic tasks



Complexity for asynchronous periodic tasks



Complexity for asynchronous periodic tasks

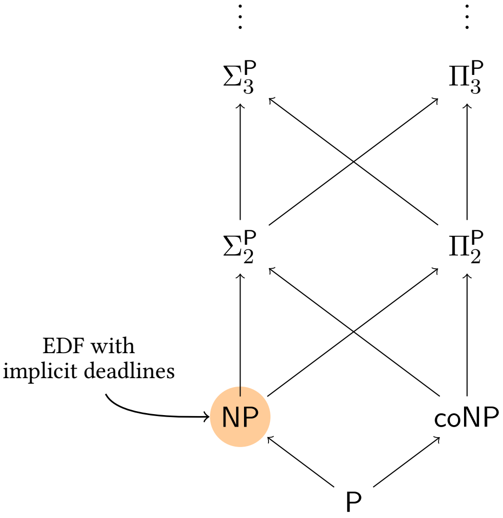


Complexity for sporadic / synchronous periodic tasks

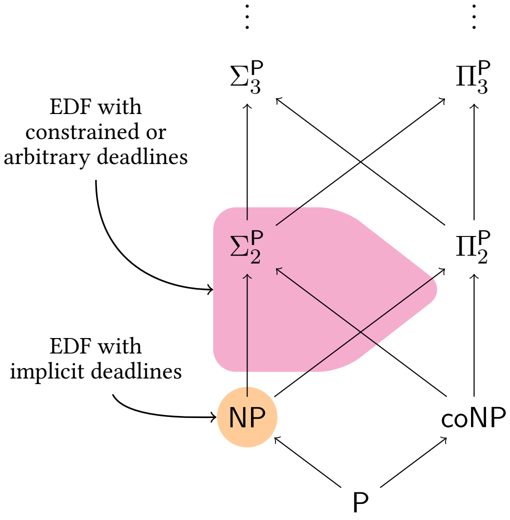
Complexity for sporadic / synchronous periodic tasks

$$
\begin{tikzcd}
\vdots & & \vdots \\
\Sigma_3^{\mathsf{P}} & & \Pi_3^{\mathsf{P}} \\
\Sigma_2^{\mathsf{P}} \arrow[u] \arrow[urr] & & \Pi_2^{\mathsf{P}} \arrow[u] \arrow[ull] \\
\mathsf{NP} \arrow[u] \arrow[urr] & & \mathsf{coNP} \arrow[u] \arrow[ull] \\
& \mathsf{P} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

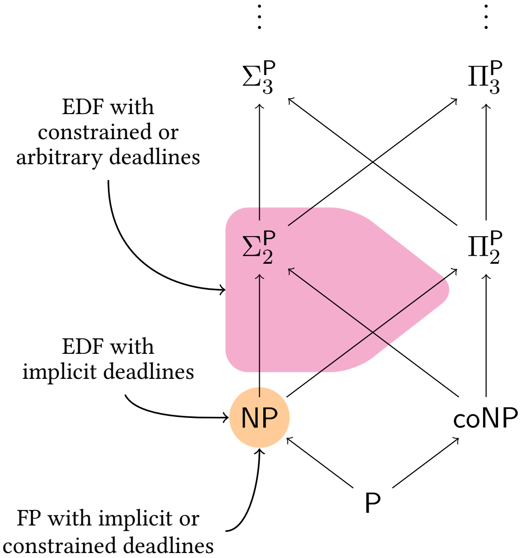
Complexity for sporadic / synchronous periodic tasks



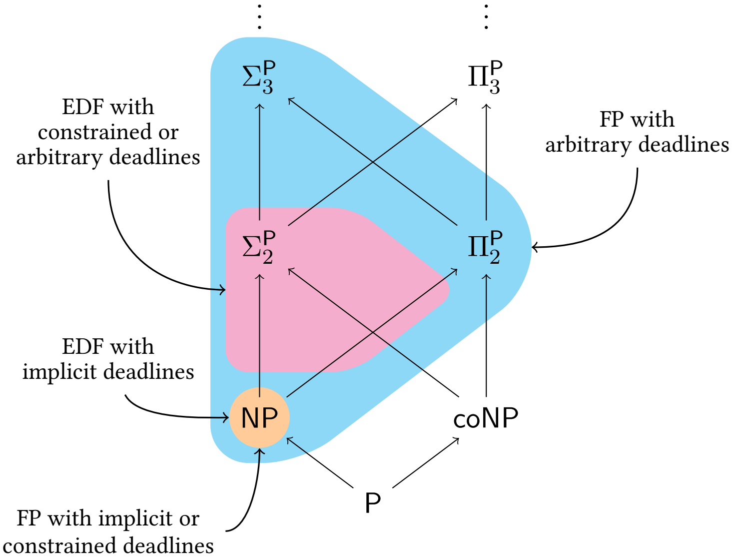
Complexity for sporadic / synchronous periodic tasks



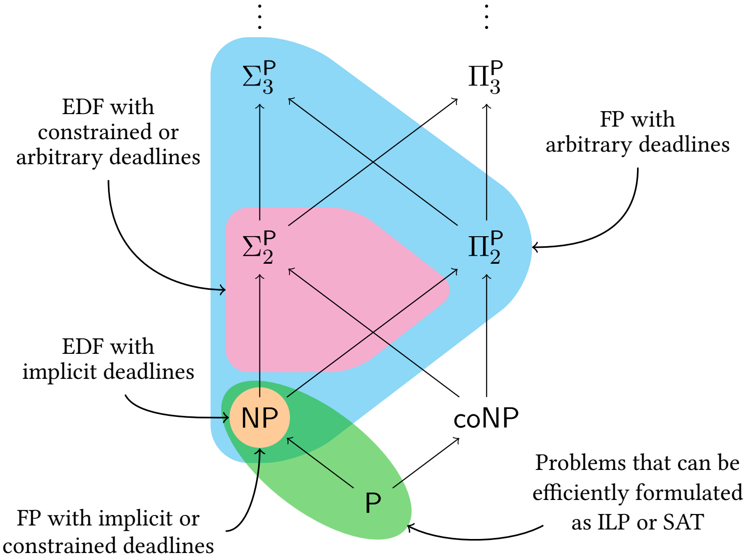
Complexity for sporadic / synchronous periodic tasks



Complexity for sporadic / synchronous periodic tasks



Complexity for sporadic / synchronous periodic tasks



Conclusions

New complexity bounds for partitioned schedulability.

Conclusions

New complexity bounds for partitioned schedulability.

- Some problems are exactly pinpointed.

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.
- *Interesting open problems!* ☺

†: Unless the polynomial hierarchy collapses

$\forall$Thank you!

$\Diamond$

$\exists$Questions?