SCHEDULING THEORY, PART 2
Periodic tasks

Pontus Ekberg

UPPSALA UNIVERSITY

2018-09-17
Today’s topic

Scheduling a finite set of jobs $j_i = (A_i, C_i, D_i) \in \mathbb{N}^3$, where

- $A_i$ is the arrival time (or release time).
- $C_i$ is the worst-case execution time (WCET), and
- $D_i$ is the deadline.

Last time

Scheduling a collection of tasks, each of which generates an unbounded sequence of jobs.
Today’s topic

Scheduling a finite set of jobs $j_i = (A_i, C_i, D_i) \in \mathbb{N}^3$, where

- $A_i$ is the arrival time (or release time).
- $C_i$ is the worst-case execution time (WCET), and
- $D_i$ is the deadline.

Last time

Scheduling a collection of tasks, each of which generates an unbounded sequence of jobs.
The (synchronous) periodic task

A (synchronous) periodic task $\tau_i$ is given by a triple $(C_i, D_i, T_i) \in \mathbb{N}^3$, where

- $C_i$ is the worst-case execution time (WCET),
- $D_i$ is the relative deadline, and
- $T_i$ is the period.
The (synchronous) periodic task

A (synchronous) periodic task $\tau_i$ is given by a triple $(C_i, D_i, T_i) \in \mathbb{N}^3$, where

- $C_i$ is the worst-case execution time (WCET),
- $D_i$ is the relative deadline, and
- $T_i$ is the period.

$$\tau_i = (C_i, D_i, T_i) = (3, 6, 10)$$
The (synchronous) periodic task

A (synchronous) periodic task $\tau_i$ is given by a triple $(C_i, D_i, T_i) \in \mathbb{N}^3$, where

- $C_i$ is the worst-case execution time (WCET),
- $D_i$ is the relative deadline, and
- $T_i$ is the period.

$$\tau_i = (C_i, D_i, T_i) = (3, 6, 10)$$
A (synchronous) periodic task $\tau_i$ is given by a triple $(C_i, D_i, T_i) \in \mathbb{N}^3$, where

- $C_i$ is the worst-case execution time (WCET),
- $D_i$ is the relative deadline, and
- $T_i$ is the period.

$\tau_i = (C_i, D_i, T_i) = (3, 6, 10)$
A (synchronous) periodic task $\tau_i$ is given by a triple $(C_i, D_i, T_i) \in \mathbb{N}^3$, where

- $C_i$ is the worst-case execution time (WCET),
- $D_i$ is the relative deadline, and
- $T_i$ is the period.
A (synchronous) periodic task $\tau_i$ is given by a triple $(C_i, D_i, T_i) \in \mathbb{N}^3$, where

- $C_i$ is the worst-case execution time (WCET),
- $D_i$ is the relative deadline, and
- $T_i$ is the period.

$\tau_i = (C_i, D_i, T_i) = (3, 6, 10)$
A (synchronous) periodic task $\tau_i$ is given by a triple $(C_i, D_i, T_i) \in \mathbb{N}^3$, where

- $C_i$ is the worst-case execution time (WCET),
- $D_i$ is the relative deadline, and
- $T_i$ is the period.

![Diagram showing the relationship between $C_i$, $D_i$, and $T_i$ over time.]

$\tau_i = (C_i, D_i, T_i) = (3, 6, 10)$
A (synchronous) periodic task $\tau_i$ is given by a triple $(C_i, D_i, T_i) \in \mathbb{N}^3$, where

- $C_i$ is the worst-case execution time (WCET),
- $D_i$ is the relative deadline, and
- $T_i$ is the period.
Scheduling a collection of periodic tasks

Given a (multi-)set $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ of $n$ periodic tasks, find a schedule where all jobs generated by those tasks meet their deadlines.
Scheduling a collection of periodic tasks

The problem

Given a (multi-)set $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ of $n$ periodic tasks, find a schedule where all jobs generated by those tasks meet their deadlines.

Assumptions

- All jobs are independent
- A single processor
- Fully preemptive scheduling
Three classes of task sets

We say that a task set $\mathcal{T}$ has

- **implicit deadlines** if $D_i = T_i$ for all $\tau_i \in \mathcal{T}$,
- **constrained deadlines** if $D_i \leq T_i$ for all $\tau_i \in \mathcal{T}$,
- **arbitrary deadlines** if $D_i$ and $T_i$ are unrelated.
We say that a task set $\mathcal{T}$ has

- **implicit deadlines** if $D_i = T_i$ for all $\tau_i \in \mathcal{T}$,
- **constrained deadlines** if $D_i \leq T_i$ for all $\tau_i \in \mathcal{T}$,
- **arbitrary deadlines** if $D_i$ and $T_i$ are unrelated.

We usually take $C_i \leq D_i$ and $C_i \leq T_i$ as unspoken constraints. (Why?)
Challenge

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]

Note: The tasks are \textit{synchronous}, so their first jobs are all released at the same time point (say, at time zero).
A solution?

Earliest Deadline First (EDF)

Scheduling rule:
Choose among the ready jobs to execute the job with the earliest absolute deadline (ties broken arbitrarily).
Let’s try it...

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it...

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

\[ T = f(1, 4, 4), (2, 3, 5), (3, 9, 10) \]
Let’s try it…

\[ T = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

\[ T = f(1, 4, 4); (2, 3, 5); (3, 9, 10) \]

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]

Schedule these tasks

\[ H_p(T) = \text{the LCM of the periods} \]
Let’s try it...

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]

Schedule these tasks:

- HP (T): the LCM of the periods
Let’s try it…

\[ T = f(1, 4, 4), (2, 3, 5), (3, 9, 10) \]

Schedule these tasks
Let’s try it...

\[ T = f(1, 4, 4), \ (2, 3, 5), \ (3, 9, 10) \]

Schedule these tasks

\[ \mathcal{S} = \{(1, 4, 4), \ (2, 3, 5), \ (3, 9, 10)\} \]
Let’s try it...

\[ T = f(1, 4, 4), (2, 3, 5), (3, 9, 10) \]

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

\[ T = f(1, 4, 4); (2, 3, 5); (3, 9, 10) \]

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it...

Schedule these tasks

\[ \mathcal{J} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

\[ T = f(1,4,4), (2,3,5), (3,9,10) \]

Schedule these tasks
Let’s try it…

\[ T = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

\[ T = f(1, 4, 4); (2, 3, 5); (3, 9, 10) \]

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it…

\[
T = (1, 4, 4), (2, 3, 5), (3, 9, 10)
\]

Schedule these tasks.
Let’s try it…

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]
Let’s try it...

\[ T = f(1, 4, 4), (2, 3, 5), (3, 9, 10) \]

Schedule these tasks

\[ \mathcal{J} = \{ (1, 4, 4), (2, 3, 5), (3, 9, 10) \} \]
Let’s try it...

$T = \{ (1, 4, 4), (2, 3, 5), (3, 9, 10) \}$
Let’s try it…

Schedule these tasks

\[ \mathcal{T} = \{(1, 4, 4), (2, 3, 5), (3, 9, 10)\} \]

\[ HP(\mathcal{T}) = \text{the LCM of the periods} \]
A solution!

A cyclic executive

Create a *static* schedule for one hyper-period, then just repeat it over and over.
A solution!

Create a *static* schedule for one hyper-period, then just repeat it over and over.

- Simple
- Predictable
- Easy to implement

Caveat: Extra care needs to be taken with arbitrary deadlines!
A solution!

A cyclic executive

Create a *static* schedule for one hyper-period, then just repeat it over and over.

- **Simple**
- **Predictable**
- **Easy to implement**

- **Not very flexible**
- Requires strict periods
- Hyper-period can be **HUGE**

Caveat: Extra care needs to be taken with arbitrary deadlines!
A solution!

Create a static schedule for one hyper-period, then just repeat it over and over.

A cyclic executive

+ Simple
+ Predictable
+ Easy to implement

- Not very flexible
- Requires strict periods
- Hyper-period can be HUGE

Caveat: Extra care needs to be taken with arbitrary deadlines!
A more flexible approach?

Question

Do we need a static scheduling table?
A more flexible approach?

Question

Do we need a static scheduling table?

No!

We can just run our scheduling algorithm (e.g., EDF) on the fly.
On EDF

Question

Is EDF still optimal for preemptive scheduling of periodic tasks?
On EDF

Question

Is EDF still optimal for preemptive scheduling of periodic tasks?

Theorem (Dertouzos, 1973)

EDF is optimal for scheduling *any* set of independent jobs on a single preemptive processor.
On EDF

Question

Is EDF still optimal for preemptive scheduling of periodic tasks?

Theorem (Dertouzos, 1973)

EDF is optimal for scheduling any set of independent jobs on a single preemptive processor.

Yes!

EDF only cares about the currently ready jobs, and “any” means any.
THE COMPONENTS OF REAL-TIME SCHEDULING THEORY

Task models:
Formalisms to specify workload and timing constraints

Scheduling algorithms:
Run-time strategies for scheduling workload

Analysis:
Offline methods for proving timing safety
THE COMPONENTS OF REAL-TIME SCHEDULING THEORY

Task models:

Formalisms to specify workload and timing constraints

Scheduling algorithms:

Run-time strategies for scheduling workload

Analysis:

Offline methods for proving timing safety
First, a useful metric

The utilization of a task $\tau_i$ is defined as

\[ U(\tau_i) = \frac{C_i}{T_i}. \]

The utilization of a task set $\mathcal{T}$ is the sum of the individual utilizations of the tasks:

\[ U(\mathcal{T}) = \sum_{\tau_i \in \mathcal{T}} U(\tau_i) = \sum_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i}. \]
First, a useful metric

The utilization of a task $\tau_i$ is defined as

$$U(\tau_i) = \frac{C_i}{T_i}.$$

The utilization of a task set $\mathcal{T}$ is the sum of the individual utilizations of the tasks:

$$U(\mathcal{T}) = \sum_{\tau_i \in \mathcal{T}} U(\tau_i) = \sum_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i}.$$

A necessary condition: Any task set $\mathcal{T}$ is infeasible unless $U(\mathcal{T}) \leq 1$. 
The demand bound function for job sets

For a job \( j_i \) and time instants \( t_1 \) and \( t_2 \), where \( 0 \leq t_1 \leq t_2 \), let the demand bound function \( \text{dbf}(j_i, t_1, t_2) \) be defined as

\[
\text{dbf}(j_i, t_1, t_2) = \begin{cases} 
C_i, & \text{if } t_1 \leq A_i \text{ and } D_i \leq t_2 \\
0, & \text{otherwise}.
\end{cases}
\]

For a job set \( J \), let \( \text{dbf}(J, t_1, t_2) \) be defined as

\[
\text{dbf}(J, t_1, t_2) = \sum_{j_i \in J} \text{dbf}(j_i, t_1, t_2).
\]
Recall: the feasibility test from last time

The demand bound function for job sets

For a job set $\mathcal{J}$ we have

$$\text{dbf}(\mathcal{J}, t_1, t_2) = \left\{ \begin{array}{l}
\text{The sum of the execution times of the jobs in } \mathcal{J} \text{ with scheduling windows fully inside the time interval } [t_1, t_2].
\end{array} \right.$$
Recall: the feasibility test from last time

The demand bound function for job sets

For a job set $\mathcal{J}$ we have

$$\text{dbf}(\mathcal{J}, t_1, t_2) = \begin{cases} \text{The sum of the execution times of the jobs in } \mathcal{J} \text{ with scheduling windows fully inside the time interval } [t_1, t_2]. \end{cases}$$

Feasibility test

A job set $\mathcal{J}$ is feasible on a single preemptive processor iff

$$\forall t_1, t_2 \text{ such that } 0 \leq t_1 \leq t_2 : \quad \text{dbf}(\mathcal{J}, t_1, t_2) \leq t_2 - t_1.$$
Let’s just adapt it to periodic tasks!
Let’s just adapt it to periodic tasks!

For a periodic task set $\mathcal{T}$ we have

$$\text{dbf}(\mathcal{T}, t_1, t_2) = \begin{cases} & \text{The sum of the execution times of the} \\ & \text{jobs from } \mathcal{T} \text{ with scheduling windows} \\ & \text{fully inside the time interval } [t_1, t_2]. \end{cases}$$
Let’s just adapt it to periodic tasks!

The demand bound function for periodic task sets

For a periodic task set $\mathcal{T}$ we have

$$\text{dbf}(\mathcal{T}, t_1, t_2) = \begin{cases} \text{The sum of the execution times of the} \\ \text{jobs from } \mathcal{T} \text{ with scheduling windows} \\ \text{fully inside the time interval } [t_1, t_2]. \end{cases}$$

Feasibility test

$\mathcal{T}$ is feasible on a single preemptive processor iff

$$\forall t_1, t_2 \text{ such that } 0 \leq t_1 \leq t_2 : \text{ dbf}(\mathcal{T}, t_1, t_2) \leq t_2 - t_1.$$
Let’s just adapt it to periodic tasks!

The demand bound function for periodic task sets

For a periodic task set $\mathcal{T}$ we have

$$\text{dbf}(\mathcal{T}, 0, t_2) = \begin{cases} \text{The sum of the execution times of the} \\ \text{jobs from } \mathcal{T} \text{ with scheduling windows} \\ \text{fully inside the time interval } [0, t_2]. \end{cases}$$

Feasibility test

$\mathcal{T}$ is feasible on a single preemptive processor iff

$$\forall t_2, \text{ such that } 0 \leq t_2 : \quad \text{dbf}(\mathcal{T}, 0, t_2) \leq t_2 - 0.$$
Let’s just adapt it to periodic tasks!

The demand bound function for periodic task sets

For a periodic task set $\mathcal{T}$ we have

$$dbf(\mathcal{T}, t) = \begin{cases} \text{The sum of the execution times of the} \\ \text{jobs from } \mathcal{T} \text{ with scheduling windows} \\ \text{fully inside the time interval } [0, t]. \end{cases}$$

Feasibility test

$\mathcal{T}$ is feasible on a single preemptive processor iff

$$\forall t, \text{ such that } 0 \leq t : \quad dbf(\mathcal{T}, t) \leq t.$$
How to compute \( \text{dbf}(\mathcal{T}, t) \)?

\[
\text{dbf}(i; t) = \max(0; \lfloor t - D_i - T_i \rfloor + 1) \\
\text{dbf}(T; t) = \sum_i \text{dbf}(i; t)
\]

\((C_i, D_i, T_i)\)
How to compute $\text{dbf}(\mathcal{T}, t)$?

$$(C_i, D_i, T_i)$$
How to compute $\text{dbf}(\mathcal{T}, t)$?

\begin{align*}
\text{dbf}(i; t) &= \max(0; \lfloor t - D_i T_i \rfloor + 1) \\
\text{dbf}(T; t) &= \sum_{i=1}^{n} \text{dbf}(i; t)
\end{align*}
How to compute $\text{dbf} (\mathcal{I}, t)$?

$$
\text{dbf} (i; t) = \max \left( 0; \lfloor t - D_i T_i \rfloor + 1 \right)
$$

$$
\text{dbf} (T; t) = \sum_{i} \text{dbf} (i; t)
$$

$(C_i, D_i, T_i)$
How to compute \( \text{dbf}(\mathcal{T}, t) \)?

\[
\text{dbf}(i; t) = \max(0; \lfloor t \rfloor + 1) \\
\text{dbf}(T; t) = \sum_{i} 2^\lfloor T_{\text{dbf}}(i; t) \rfloor
\]
How to compute $\text{dbf}(\mathcal{T}, t)$?

\[ \text{dbf}(\tau_i, t) = \max \left( 0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) \cdot C_i \]
How to compute $\text{dbf}(\mathcal{T}, t)$?

$$
\text{dbf}(\tau_i, t) = \max \left( 0, \left\lfloor \frac{t-D_i}{T_i} \right\rfloor + 1 \right) \cdot C_i
$$

$$
\text{dbf}(\mathcal{T}, t) = \sum_{\tau_i \in \mathcal{T}} \text{dbf}(\tau_i, t)
$$
Visualizing the feasibility test
Visualizing the feasibility test

\[ t = \frac{1}{\dbf(T; t)} \quad \text{such that} \quad 0 \leq t \]
Visualizing the feasibility test

\[ t = \sum_{i=1}^{n} dbf(T; t) \leq t \]
Visualizing the feasibility test

\[ \forall t, \text{ such that } 0 \leq t : \quad \text{dbf}(\mathcal{I}, t) \leq t \]
\textbf{Visualizing the feasibility test}

\[ \forall t, \text{ such that } 0 \leq t : \quad \text{dbf}(\mathcal{I}, t) \leq t \]
Visualizing the feasibility test

∀t, such that 0 ≤ t : dbf(\mathcal{J}, t) ≤ t
A (synchronous) periodic task set $\mathcal{T}$ is feasible on a single preemptive processor iff

$$\forall t, \text{ such that } 0 \leq t : \ dbf(\mathcal{T}, t) \leq t.$$
A (synchronous) periodic task set \( \mathcal{T} \) is feasible on a single preemptive processor iff

\[
\forall t, \text{ such that } 0 \leq t : \quad \text{dbf}(\mathcal{T}, t) \leq t.
\]

Check *all* positive values for \( t \)!!
A (synchronous) periodic task set $\mathcal{T}$ is feasible on a single preemptive processor iff

$$\forall t, \text{ such that } 0 \leq t : \ dbf(\mathcal{T}, t) \leq t.$$ 

Check *all* positive values for $t$?!

If $U(\mathcal{T}) \leq 1$, we don’t have to check $t \geq HP(\mathcal{T}) + \max_{\tau_i \in \mathcal{T}} D_i$!

(Why?)
A (synchronous) periodic task set $\mathcal{T}$ is feasible on a single preemptive processor iff $U(\mathcal{T}) \leq 1$ and

$$\forall t, \text{ such that } 0 \leq t \leq HP(\mathcal{T}) + \max_{\tau_i \in \mathcal{T}} D_i : \ dbf(\mathcal{T}, t) \leq t.$$
A (synchronous) periodic task set $\mathcal{T}$ is feasible on a single preemptive processor iff $U(\mathcal{T}) \leq 1$ and

$$\forall t, \text{ such that } 0 \leq t \leq HP(\mathcal{T}) + \max_{\tau_i \in \mathcal{T}} D_i : \text{ dbf}(\mathcal{T}, t) \leq t.$$  

Feasibility test (Baruah et al., 1990)

A (synchronous) periodic task set $\mathcal{T}$ with implicit deadlines is feasible on a single preemptive processor iff

$$U(\mathcal{T}) \leq 1.$$  

Feasibility test (Liu & Layland, 1973)
A useful trick

\[ \text{HP} \left( T \right) = \text{LCM of periods} \]

\[ \text{slope} = \sum_i C_i \cdot U \left( T \right) \]

Feasibility (EDF-sched.)

Exp. time algorithm exists

In \( \text{coNP} \)

Strongly \( \text{coNP} \)-hard (ECRTS'15)

Bounded version (\( U \left( T \right) \leq c \))

Pseudo-poly. time algorithm if \( c < 1 \)

In \( \text{coNP} \)

Weakly \( \text{coNP} \)-hard for all \( c \) (RTSS'15)
A useful trick

\[ HP(T) = \text{LCM of periods} \]

Exp. time algorithm exists in \( \text{incoNP} \) and is strongly \( \text{coNP} \)-hard (ECRTS'15)

Bounded version \( (U(T) \leq c) \) has a pseudo-poly. time algorithm if \( c < 1 \)

In \( \text{coNP} \) for all \( c \) (RTSS'15)
A useful trick

\[ \text{HP}(\mathcal{J}) = \text{LCM of periods} \]

Feasibility (EDF-sched.)
- Exp. time algorithm exists

\[ \sum_i C_i \leq U(T) \]

In coNP
- Strongly coNP-hard (ECRTS'15)
- Pseudo-poly. time algorithm if \( c < 1 \)
- In coNP
- Weakly coNP-hard for all \( c \) (RTSS'15)
A useful trick

\[ \text{Feasibility (EDF-sched.)} \]
- Exp. time algorithm exists
- In coNP

\[ \text{HP}(J) = \text{LCM of periods} \]
A useful trick

$$HP(\mathcal{J}) = \text{LCM of periods}$$

Feasibility (EDF-sched.)
- Exp. time algorithm exists
- In coNP

Bounded version ($u(\mathcal{T}) \leq c$)

Pseudo-poly. time algorithm if $c < 1$

In coNP

Weakly coNP-hard for all $c$ (RTSS'15)
A useful trick

\[ \text{slope} = U(\mathcal{T}) \]

Feasibility (EDF-sched.)
- Exp. time algorithm exists
- In coNP

\[ \text{HP}(\mathcal{T}) = \text{LCM of periods} \]
A useful trick

slope = \( U(\mathcal{J}) \)

Feasibility (EDF-sched.)

- Exp. time algorithm exists
- In coNP

\[ \text{HP}(\mathcal{J}) = \text{LCM of periods} \]
A useful trick

\[ \text{HP}(J) = \text{LCM of periods} \]

Feasibility (EDF-sched.)

- Exp. time algorithm exists
- In coNP

slope = \( U(J) \)
A USEFUL TRICK

slope = U(\mathcal{J})

Feasibility (EDF-sched.)

- Exp. time algorithm exists
- In coNP

HP(\mathcal{J}) = LCM of periods
A useful trick

\[ HP(\mathcal{J}) = \text{LCM of periods} \]

Feasibility (EDF-sched.)

- Exp. time algorithm exists
- In coNP

\[ \sum_i C_i \]

\[ \text{slope} = U(\mathcal{J}) \]

\[ \text{HP(\mathcal{J})} = \text{LCM of periods} \]
A USEFUL TRICK

\[ \text{slope} = U(\mathcal{J}) \]

Feasibility (EDF-sched.)

- Exp. time algorithm exists
- In coNP

\[ \sum_i C_i \]

\[ \frac{\sum_i C_i}{1 - U(\mathcal{J})} \]

\[ \text{HP}(\mathcal{J}) = \text{LCM of periods} \]
A useful trick

$$AHP(\mathcal{T}) = \text{LCM of periods}$$

Feasibility (EDF-sched.)
- Exp. time algorithm exists
- In coNP

Bounded version \((U(\mathcal{T}) \leq c)\)
- Pseudo-poly. time algorithm if \(c < 1\)
- In coNP

\[
\sum_i C_i \leq \frac{\sum_i C_i}{1 - U(\mathcal{T})} = \text{HP}(\mathcal{T}) = \text{LCM of periods}
\]
A useful trick

HP(T) = LCM of periods

slope = U(\mathcal{J})

Feasibility (EDF-sched.)
- Exp. time algorithm exists
- In coNP
- Strongly coNP-hard (ECRTS’15)

Bounded version \ (U(\mathcal{J}) \leq c)
- Pseudo-poly. time algorithm if c < 1
- In coNP

\[ \sum_{i} C_i \]

\[ \frac{\sum_{i} C_i}{1 - U(\mathcal{J})} \]

\[ \text{HP(\mathcal{J})} = \text{LCM of periods} \]
A USEFUL TRICK

\[ \text{slope} = U(\mathcal{J}) \]

Feasibility (EDF-sched.)
- Exp. time algorithm exists
- In coNP
- Strongly coNP-hard (ECRTS’15)

Bounded version \((U(\mathcal{J}) \leq c)\)
- Pseudo-poly. time algorithm if \(c < 1\)
- In coNP
- Weakly coNP-hard for all \(c\) (RTSS’15)
Highlights

• You can make a *static schedule* for a full hyper-period, and just repeat it over and over.

• More flexible is to run a scheduler like EDF on the fly.

• EDF is optimal also in this setting.
  • But *not* the most widely used (more on this next time).

• The dbf-based test is gives an exact feasibility condition.
  • Exponential time in general.
  • Quite fast in practice, thanks to the alternative bound.

• With implicit deadlines, $U(\mathcal{T}) \leq 1$ is an easier exact test.