# Towards a Tractable Exact Schedulability Test for Real-Time Scheduling

Artem Burmyakov

March 06, 2024

INNOPOLIS UNIVERSITY

# About Myself

| **Education** | |
|---|---|
| 2004, 2007 | BSc and MSc in Applied Math, Moscow Engineering Physics Institute |
| 2016 | PhD in Computer Science, CISTER & FEUP |

| **Relevant work experience** | |
|---|---|
| Since 2019 | Post-doctoral researcher / Assistant professor, Innopolis University, Russia |
| 2016 - 2017 | Post-doctoral researcher, Seoul National University, South Korea |
| 2006-2010 | Software developer, CERN, Switzerland |

| **Delivering courses** |
|---|
| Computer Architecture (core course) |
| Computer Networks (core course) |
| Consensus Theory and Concurrent Programming (elective course) |
| Real-Time Scheduling Theory (elective course) |

| **Research Interests** |
|---|
| Real-time multiprocessor scheduling |
| Optimization of computationally intensive algorithms |
| Concurrent programming, Static analysis of multithreaded programs |

# System Model:
# Real-Time Sporadic Task

# System Model:
# Real-Time Sporadic Task

$$\tau_i = (C_i, D_i, P_i)$$

[Liu and Layland 1973]

# System Model:
# Real-Time Sporadic Task

$$\tau_i = (C_i, D_i, P_i)$$

**Execution time**

# System Model:
# Real-Time Sporadic Task

$$\tau_i = (C_i, D_i, P_i)$$

Execution time

**Relative deadline**

[Liu and Layland 1973]

# System Model:
# Real-Time Sporadic Task

$$\tau_i = (C_i, D_i, P_i)$$

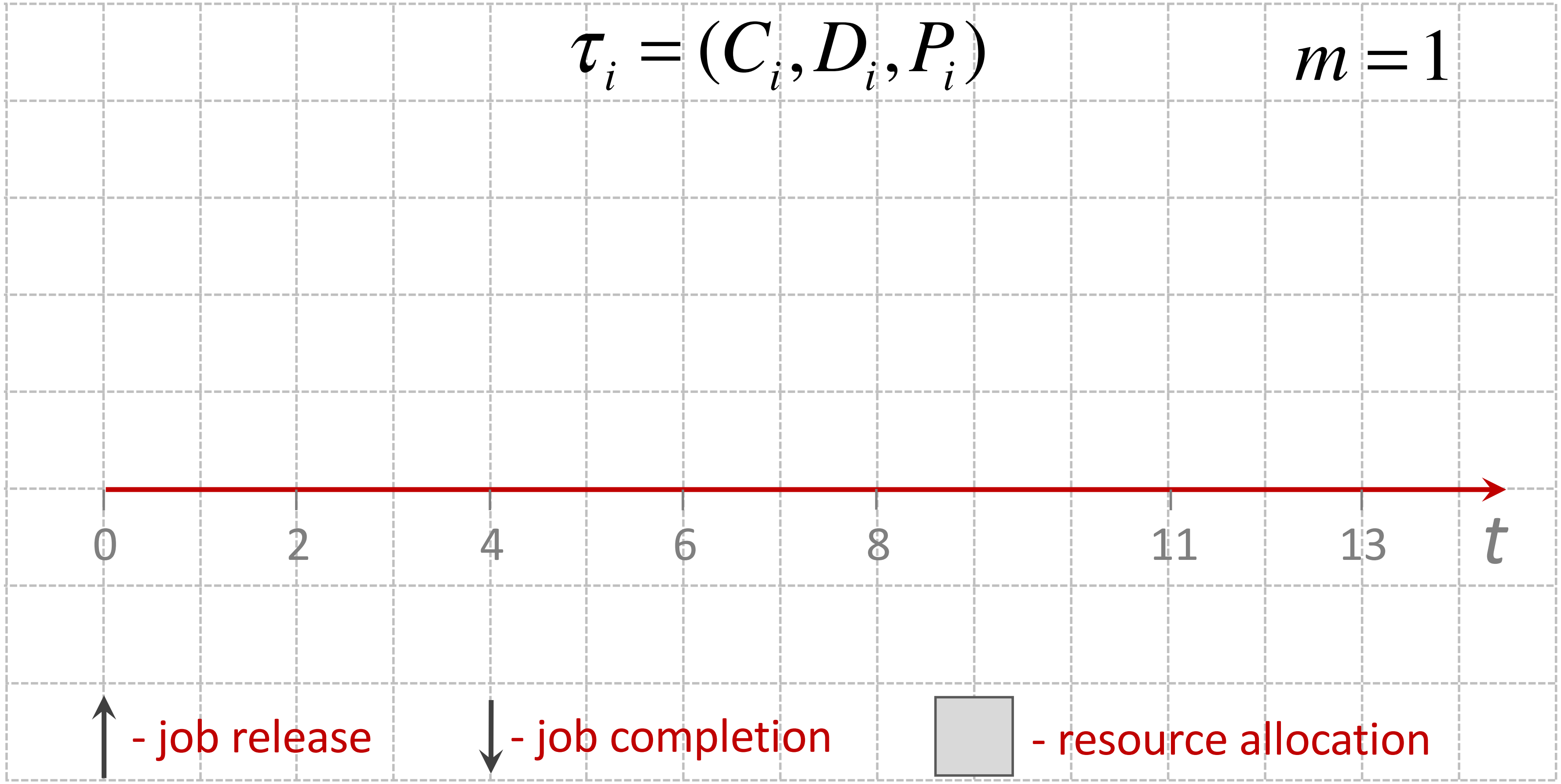
Execution time

Relative deadline

**Minimum time between job releases (or simply period)**

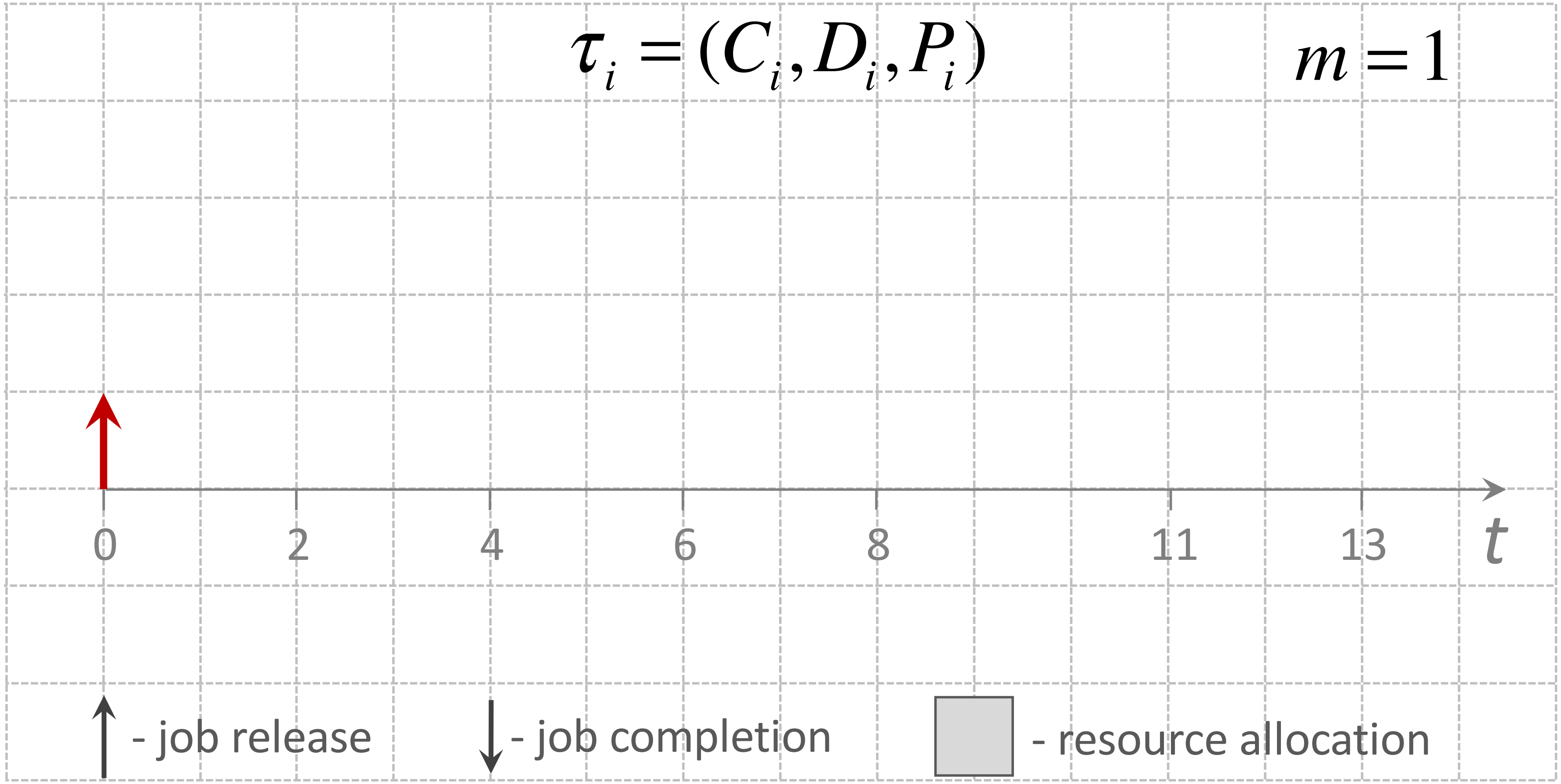
[Liu and Layland 1973]

# System Model:
# Real-Time Sporadic Task



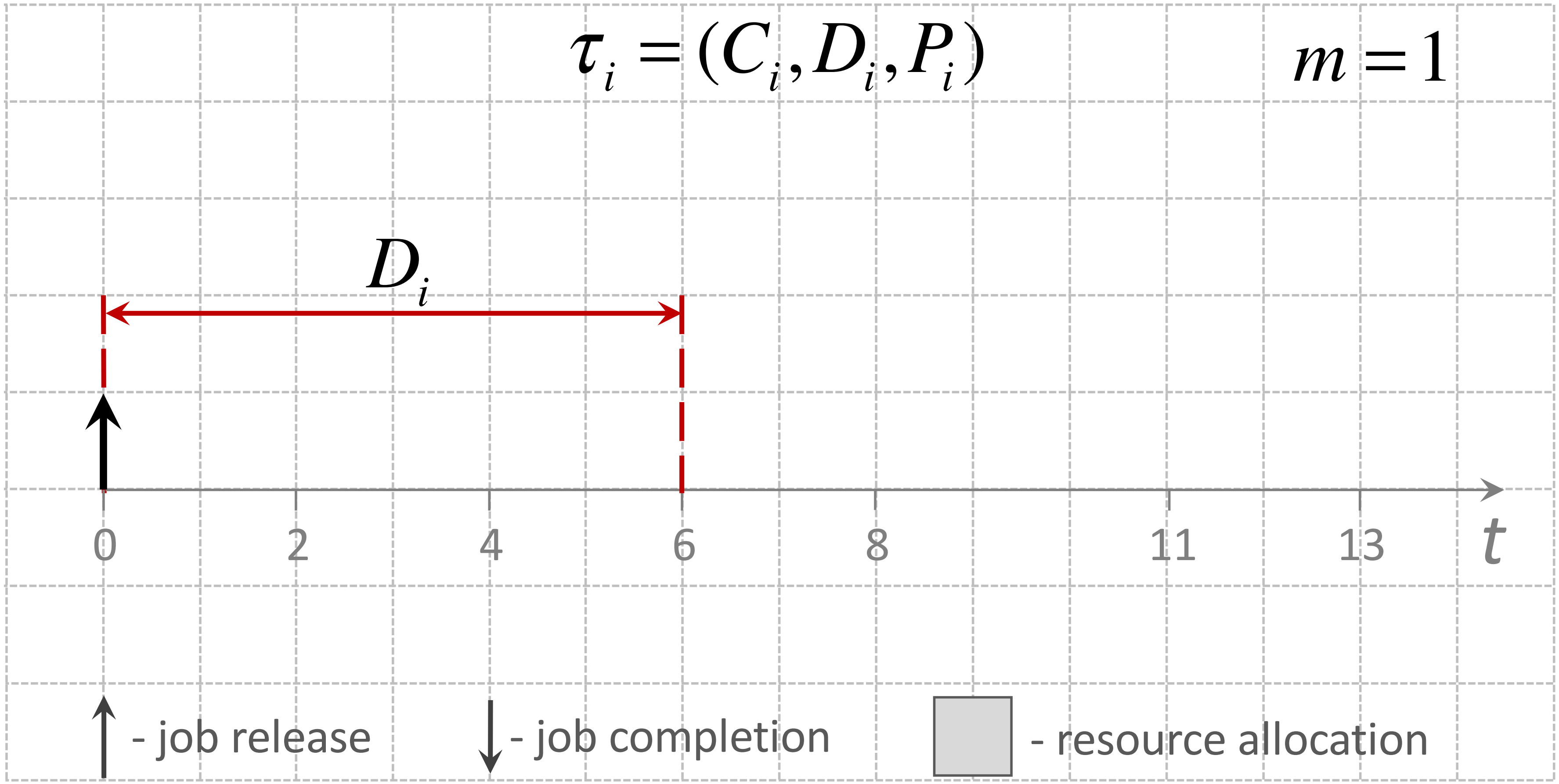

[Liu and Layland 1973]

# System Model: Real-Time Sporadic Task



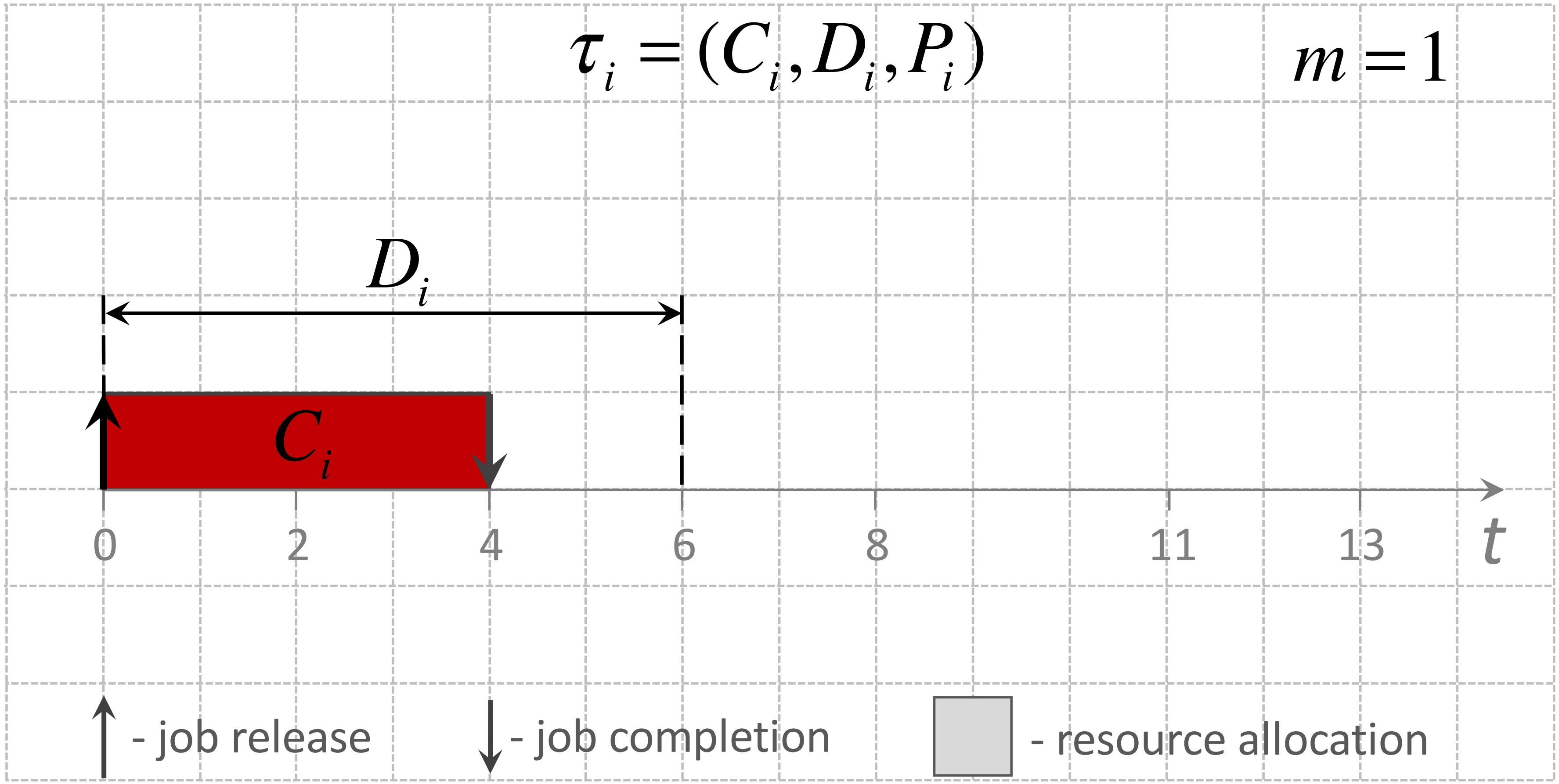

[Liu and Layland 1973]

# System Model:
# Real-Time Sporadic Task



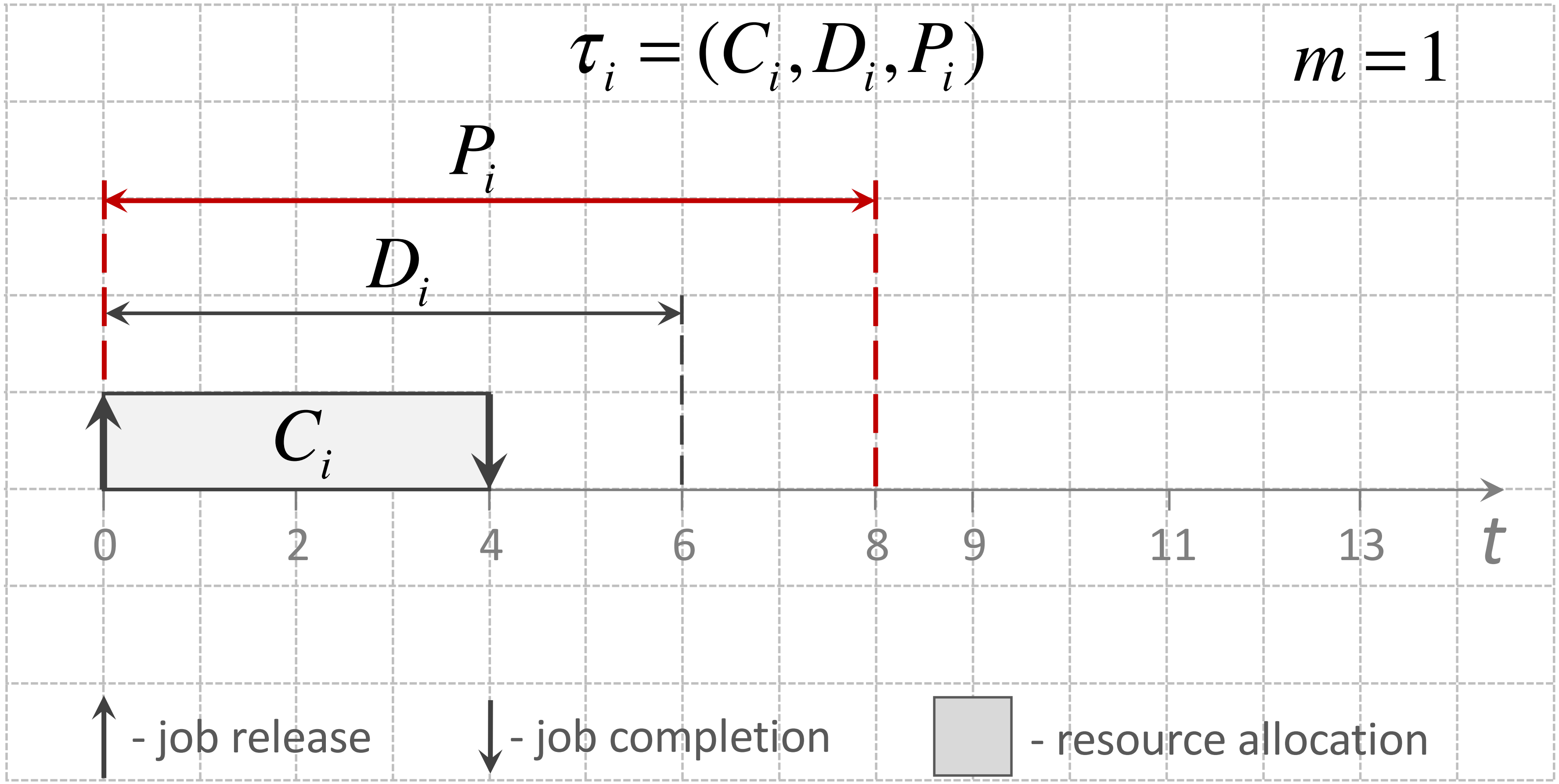

[Liu and Layland 1973]

# System Model:
# Real-Time Sporadic Task



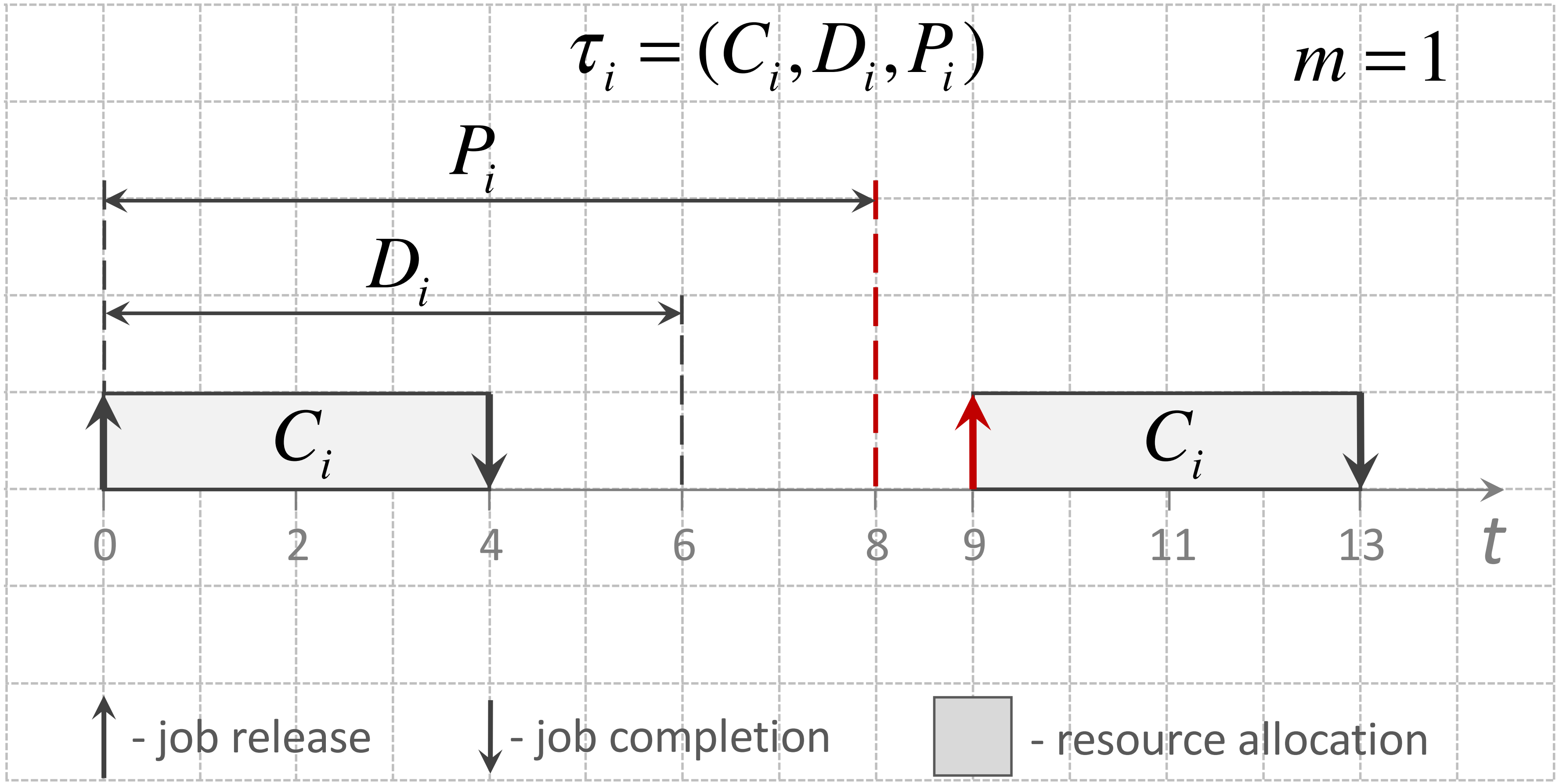

[Liu and Layland 1973]

# System Model: Real-Time Sporadic Task




[Liu and Layland 1973]

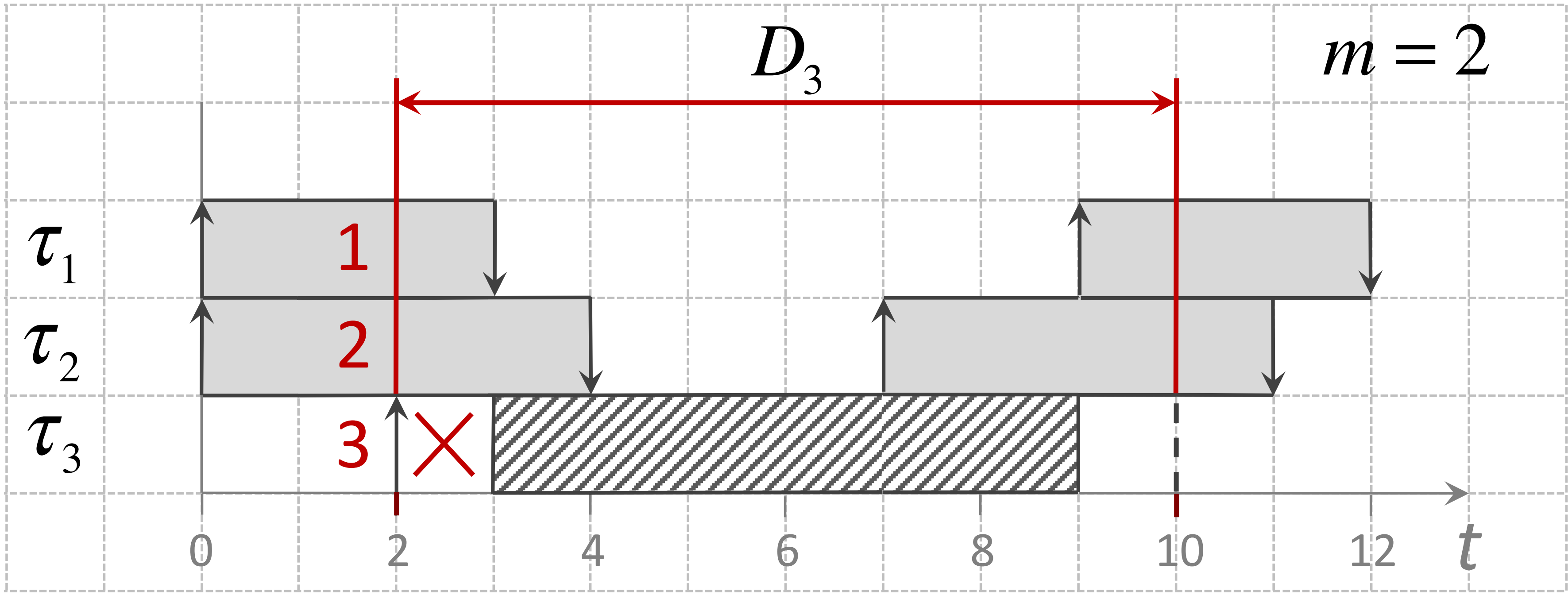
# System Model:
# Real-Time Sporadic Task

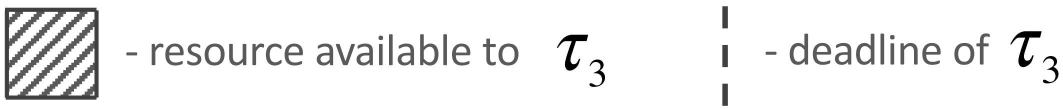



[Liu and Layland 1973]

# A Sample Global Fixed Priority (GFP) Schedule

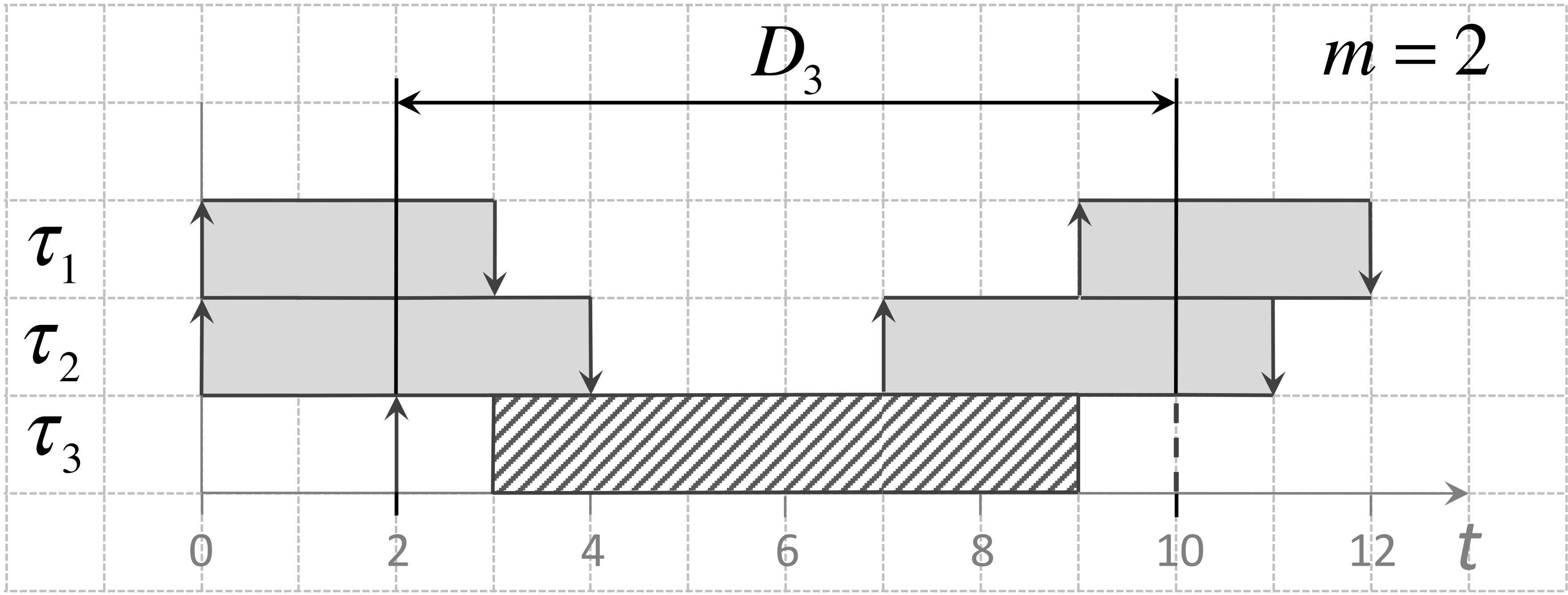
Priorities assumption: Task with a smaller index has a higher priority



# A Sample Global Fixed Priority (GFP) Schedule

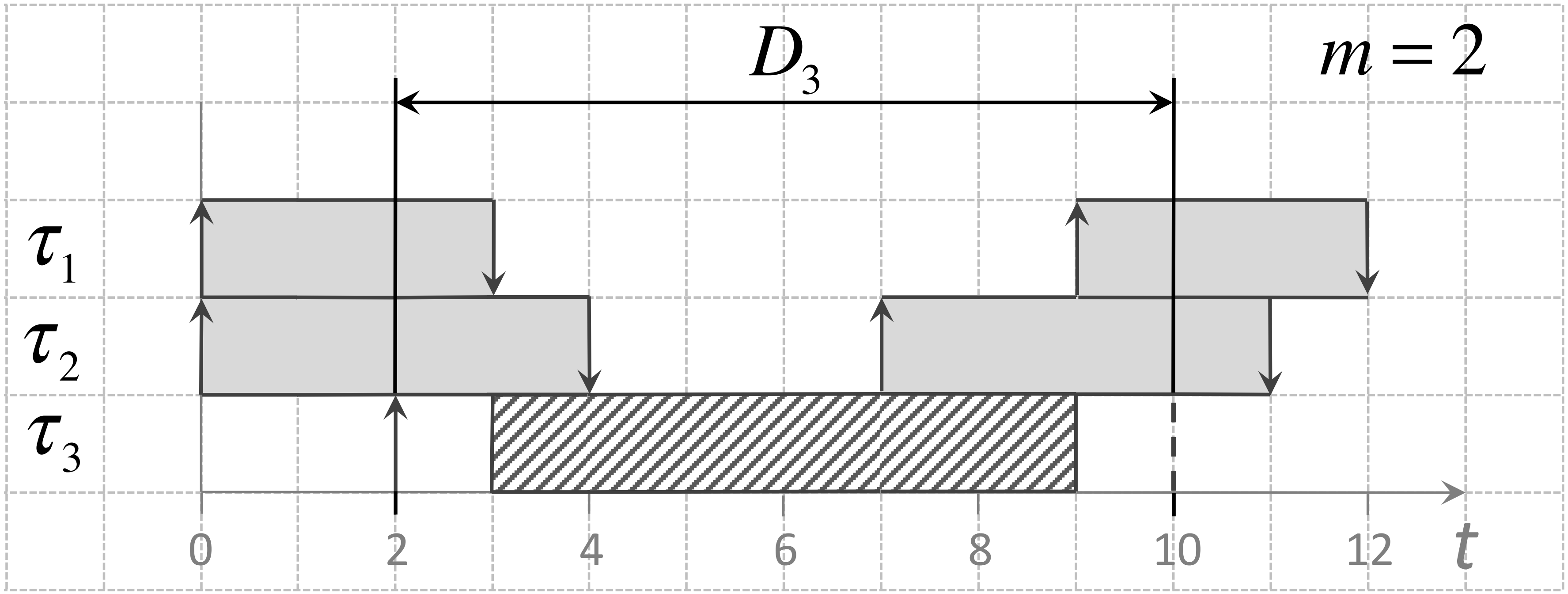
Priorities assumption: Task with a smaller index has a higher priority




**Question: Is there such a scenario, that the task $\tau_3$ misses a deadline?**

# A Sample Global Fixed Priority (GFP) Schedule

Priorities assumption: Task with a smaller index has a higher priority



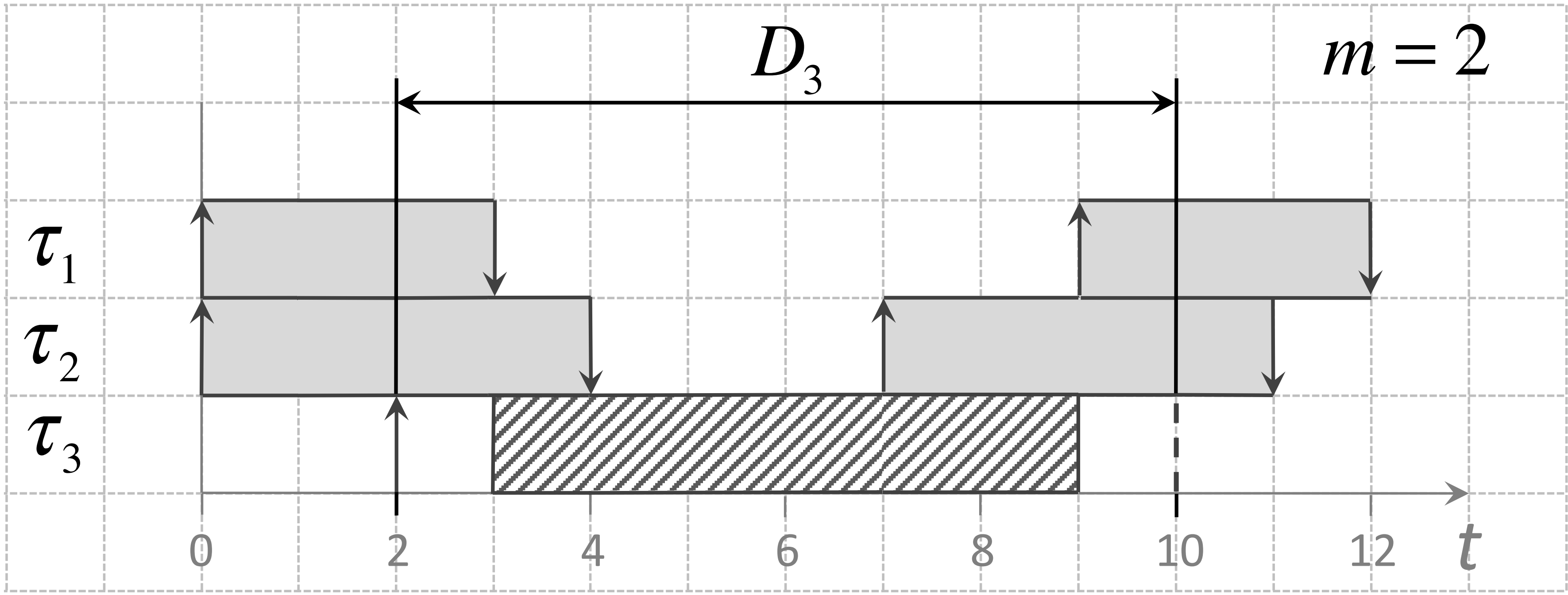

**Question: Is there such a scenario, that the task $\tau_3$ misses a deadline?**

**Possible solutions:**

- **Exact (NP-hard)**

# A Sample Global Fixed Priority (GFP) Schedule

Priorities assumption: Task with a smaller index has a higher priority




**Question: Is there such a scenario, that the task $\tau_3$ misses a deadline?**

**Possible solutions:**

- **Exact (NP-hard);**
- **Suboptimal (pessimistic)**

# Schedulability Test for Multiprocessors: NP-hard Problem

| Exact | vs. | Sufficient |
|---|---|---|
| Based on examining various (all?) possible scenarios | | Estimates the worst-case execution scenario |

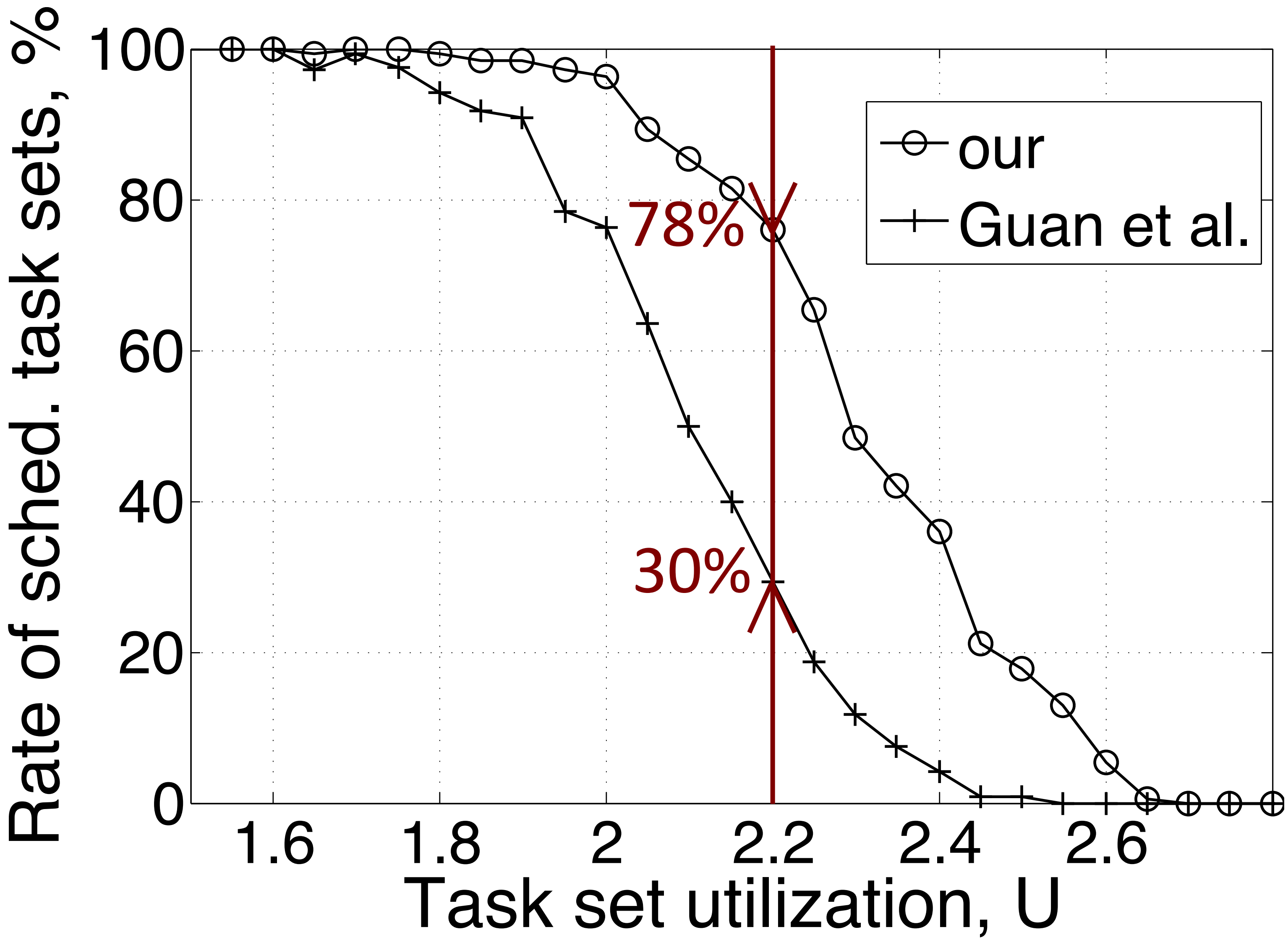
# Schedulability Test for Multiprocessors: NP-hard Problem

| Exact | vs. Sufficient |
|---|---|
| Based on examining various (all?) possible scenarios | Estimates the worst-case execution scenario |
| Exact | Pessimistic |

# Schedulability Test for Multiprocessors: NP-hard Problem

| Exact | vs. Sufficient |
|---|---|
| Based on examining various (all?) possible scenarios | Estimates the worst-case execution scenario |
| Exact | Pessimistic |
| Exponential runtime complexity | Lower (<= pseudopolynomial) runtime complexity |

# Exact vs. Sufficient Schedulability Tests




$m = 3$

$n = 7$

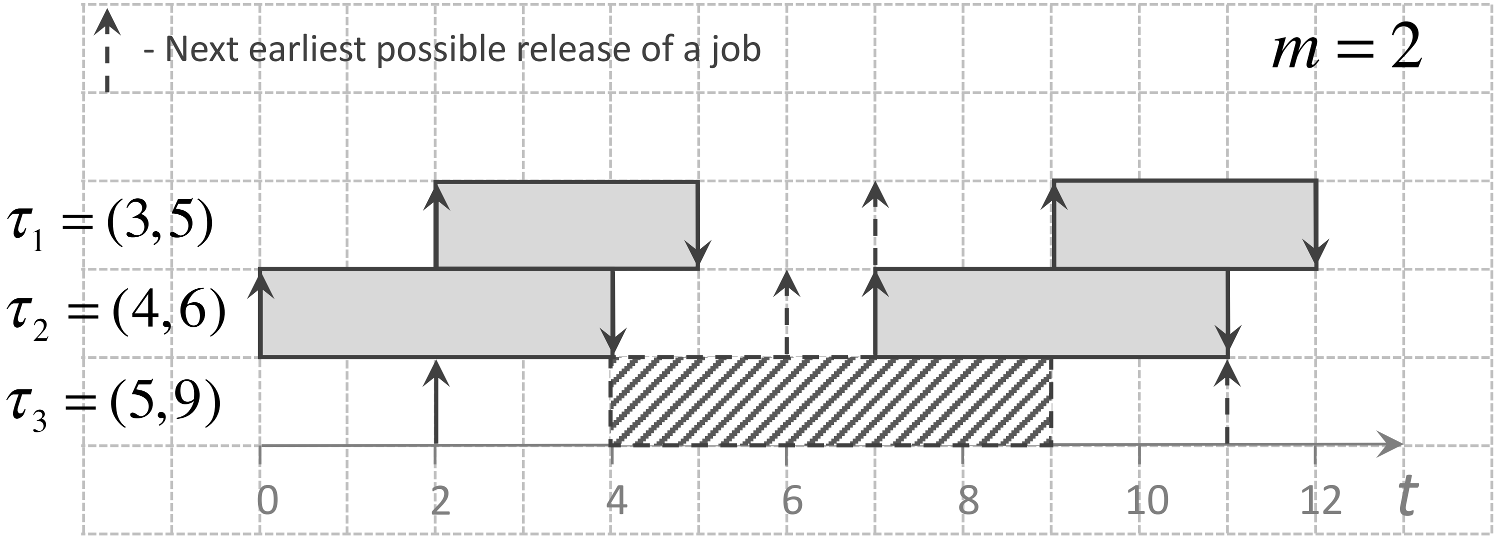
$U_{\max} = 0.6$

$P_{\max} / P_{\min} = 4$

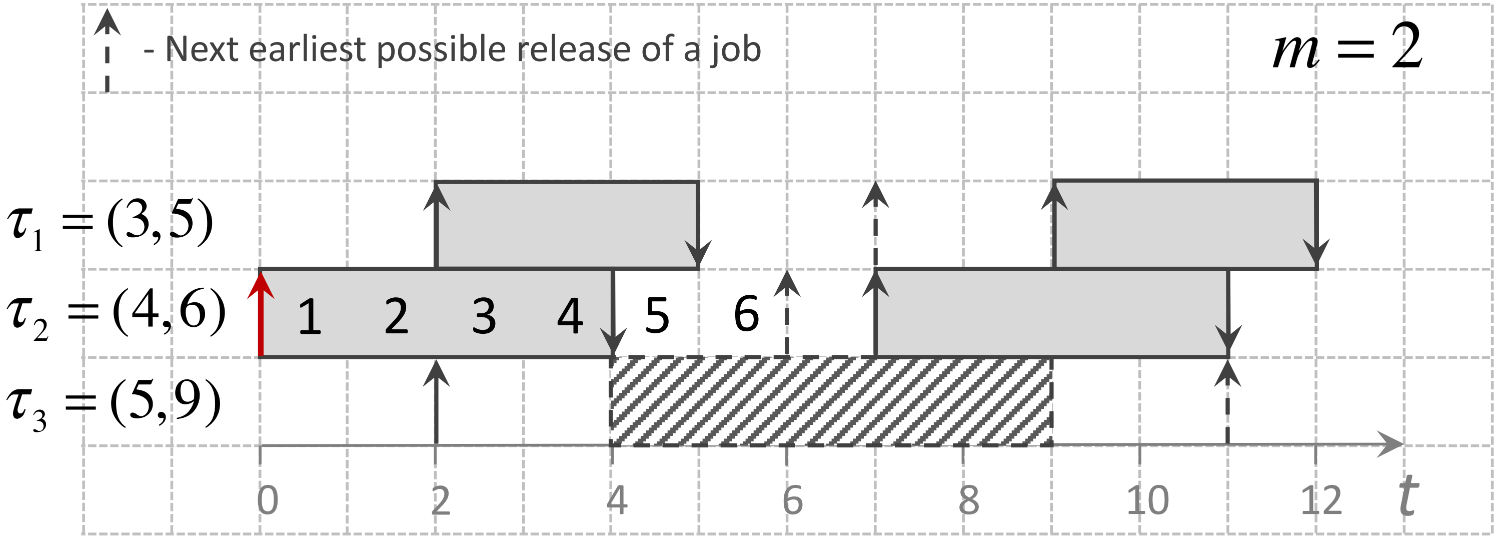
$P_{\min} \in [3,10]$

# State Transition Graph

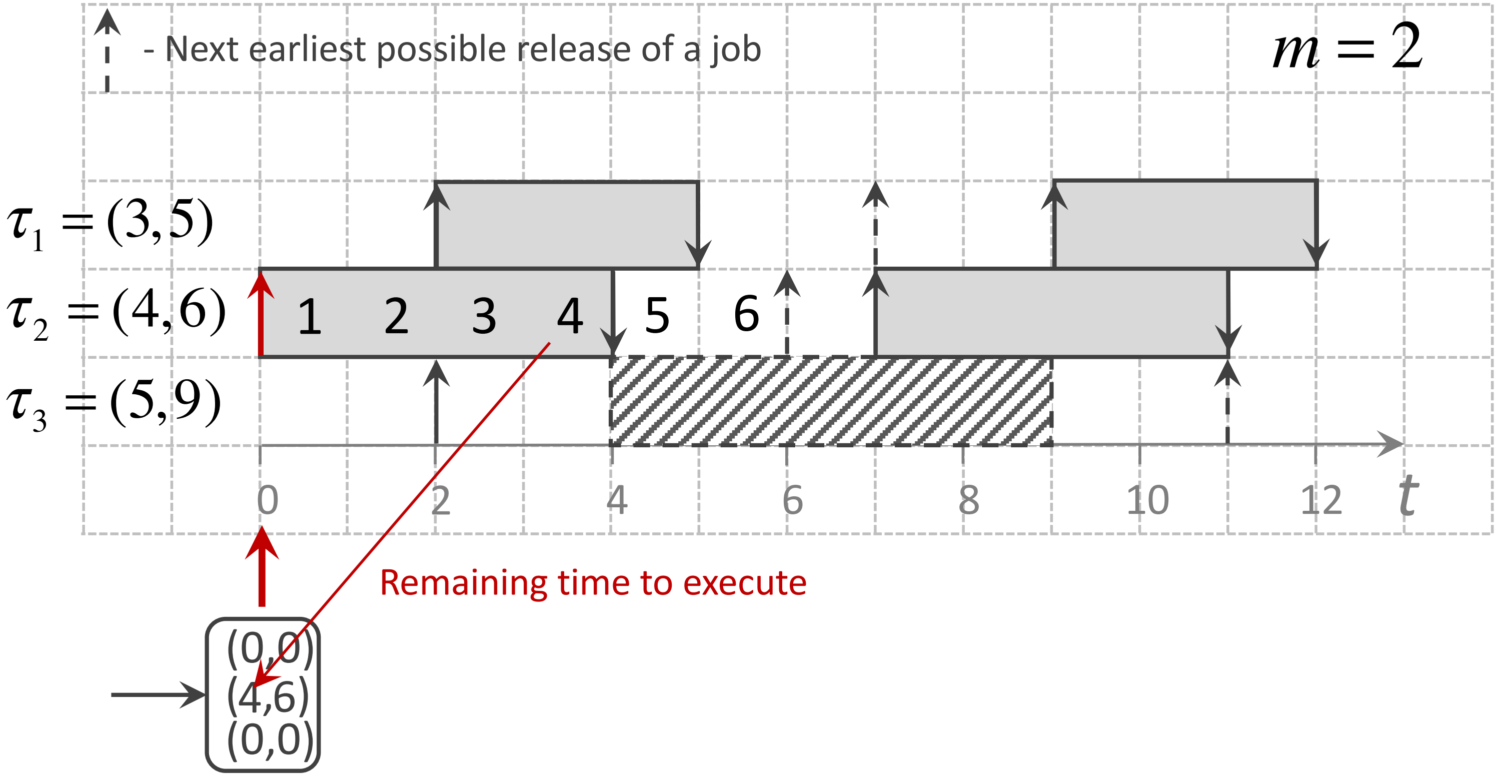
Consider the following GFP schedule:



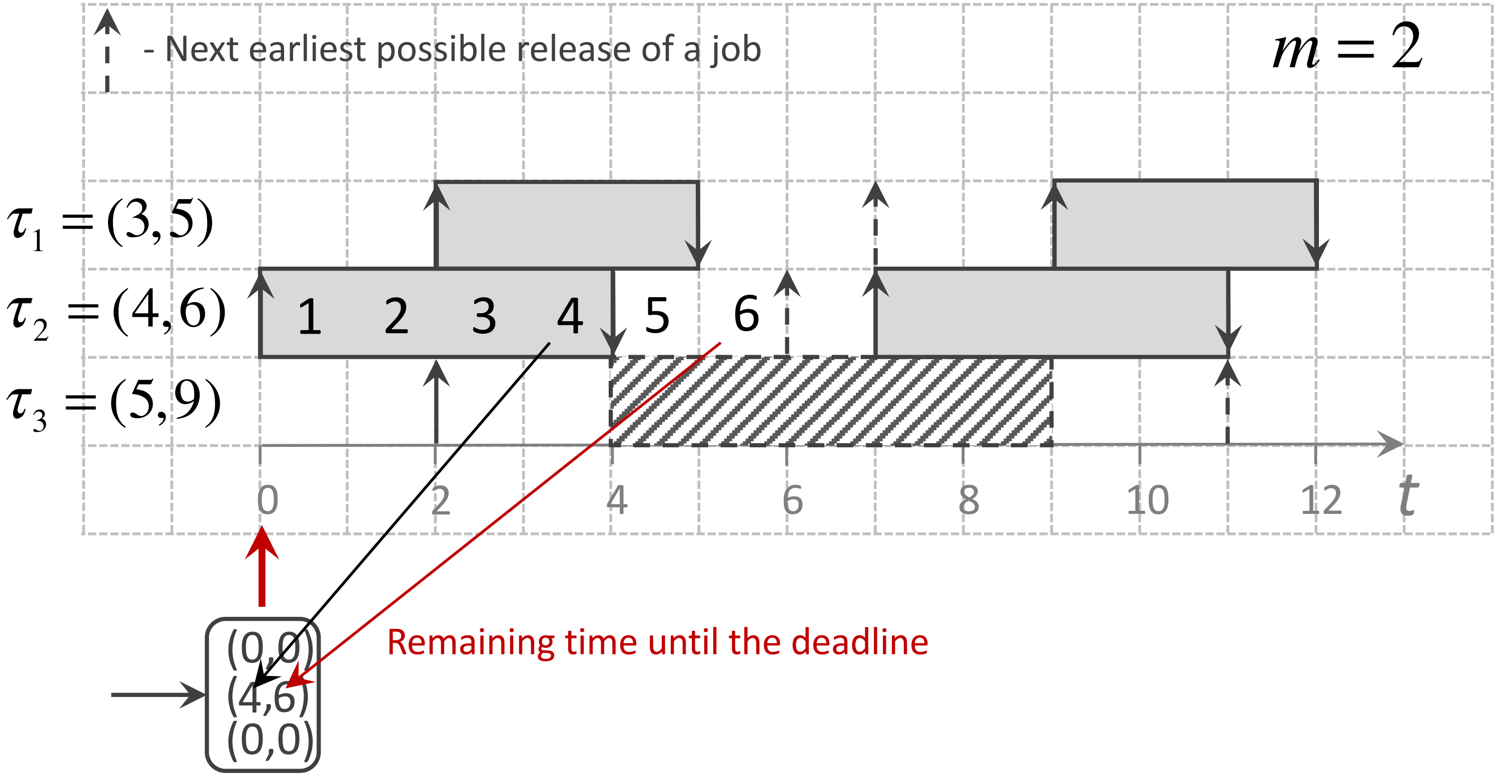
# State Transition Graph for a Schedule



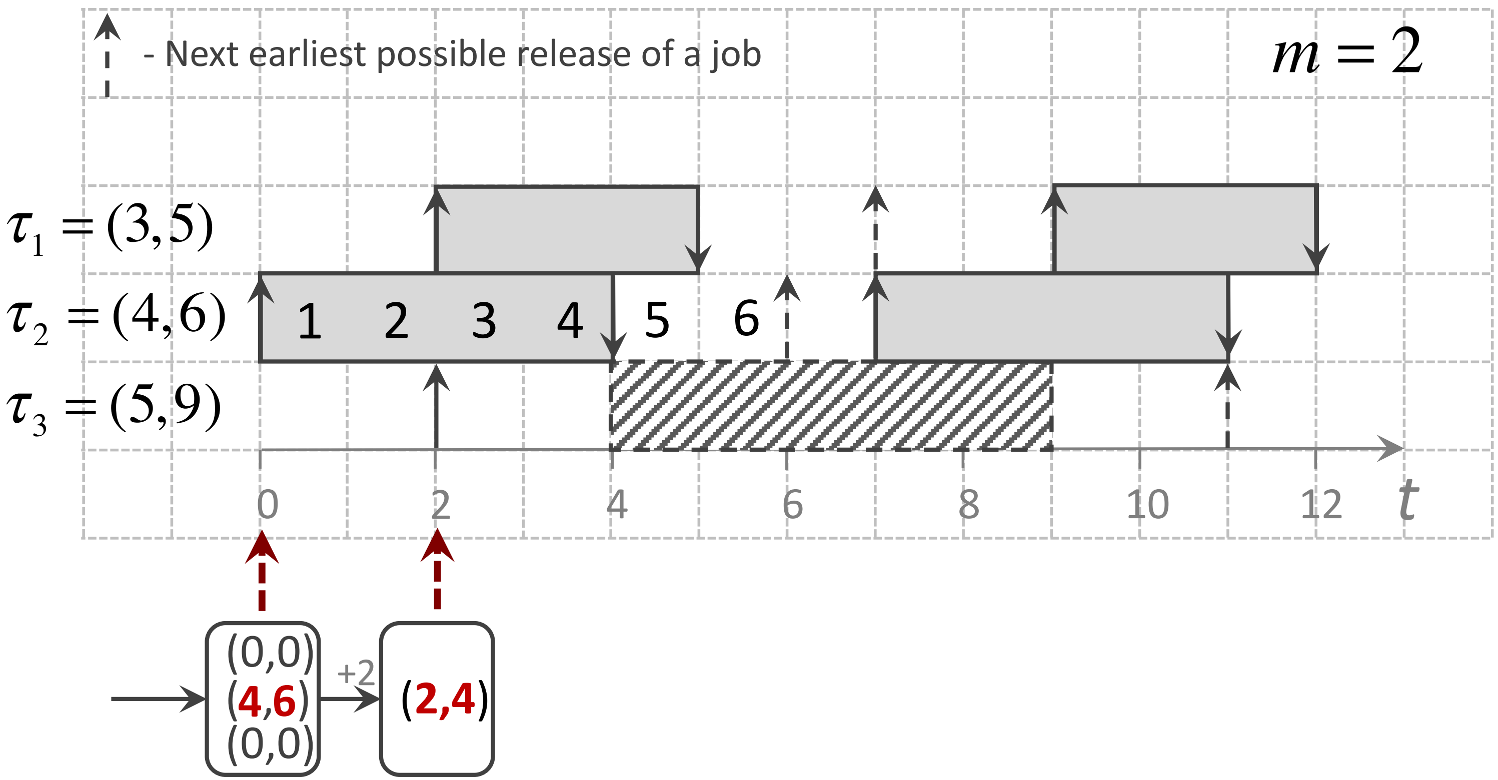
# State Transition Graph for a Schedule



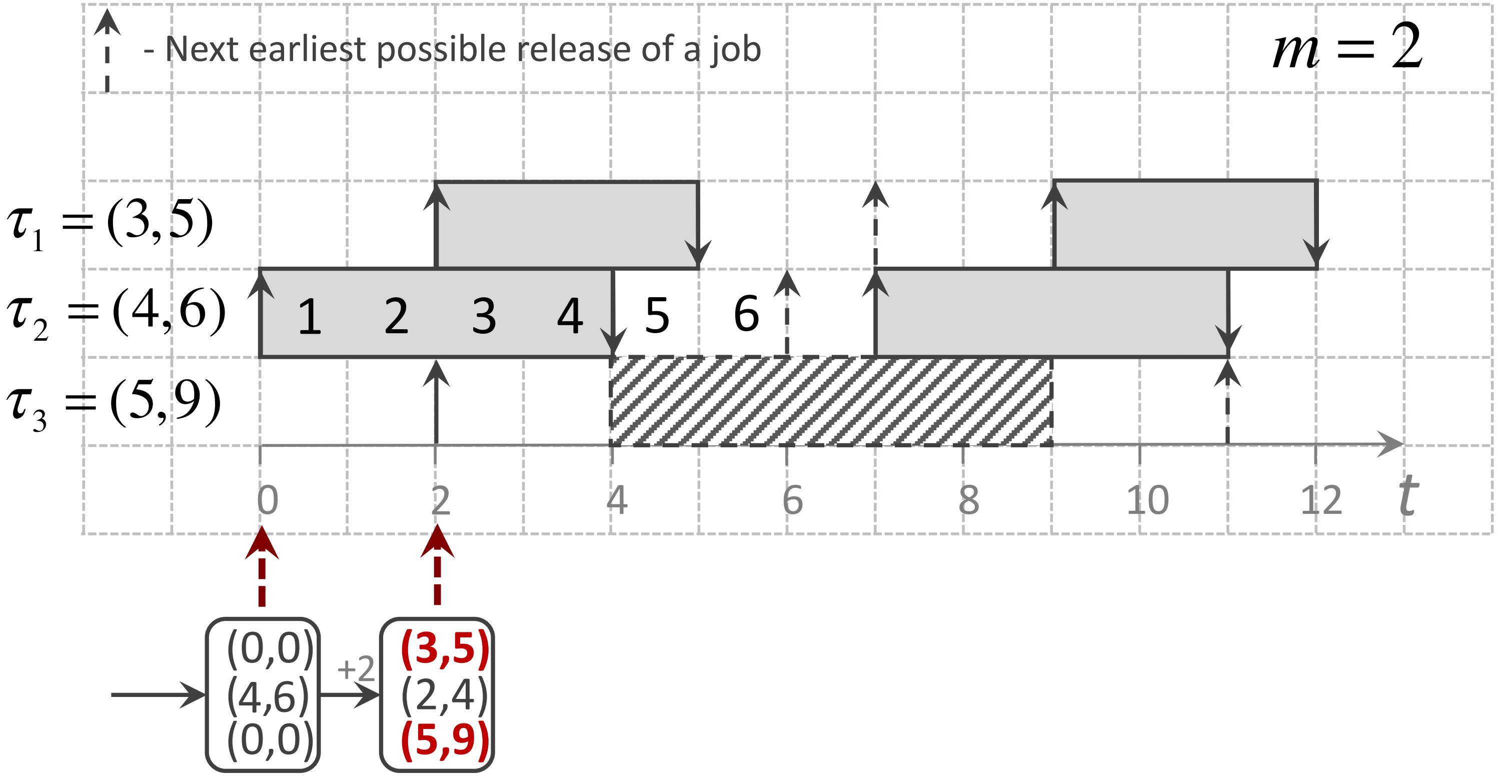
# State Transition Graph for a Schedule



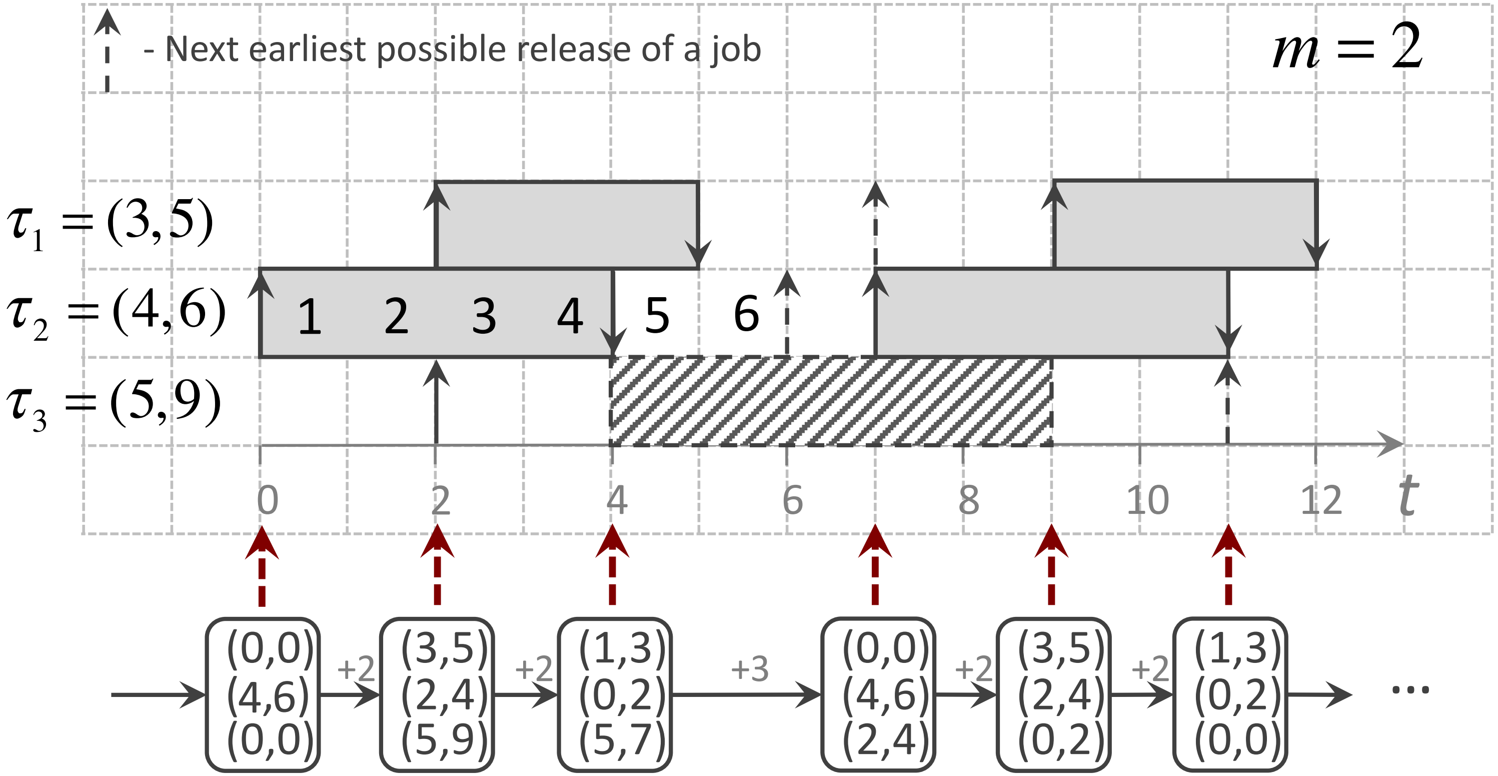
# State Transition Graph



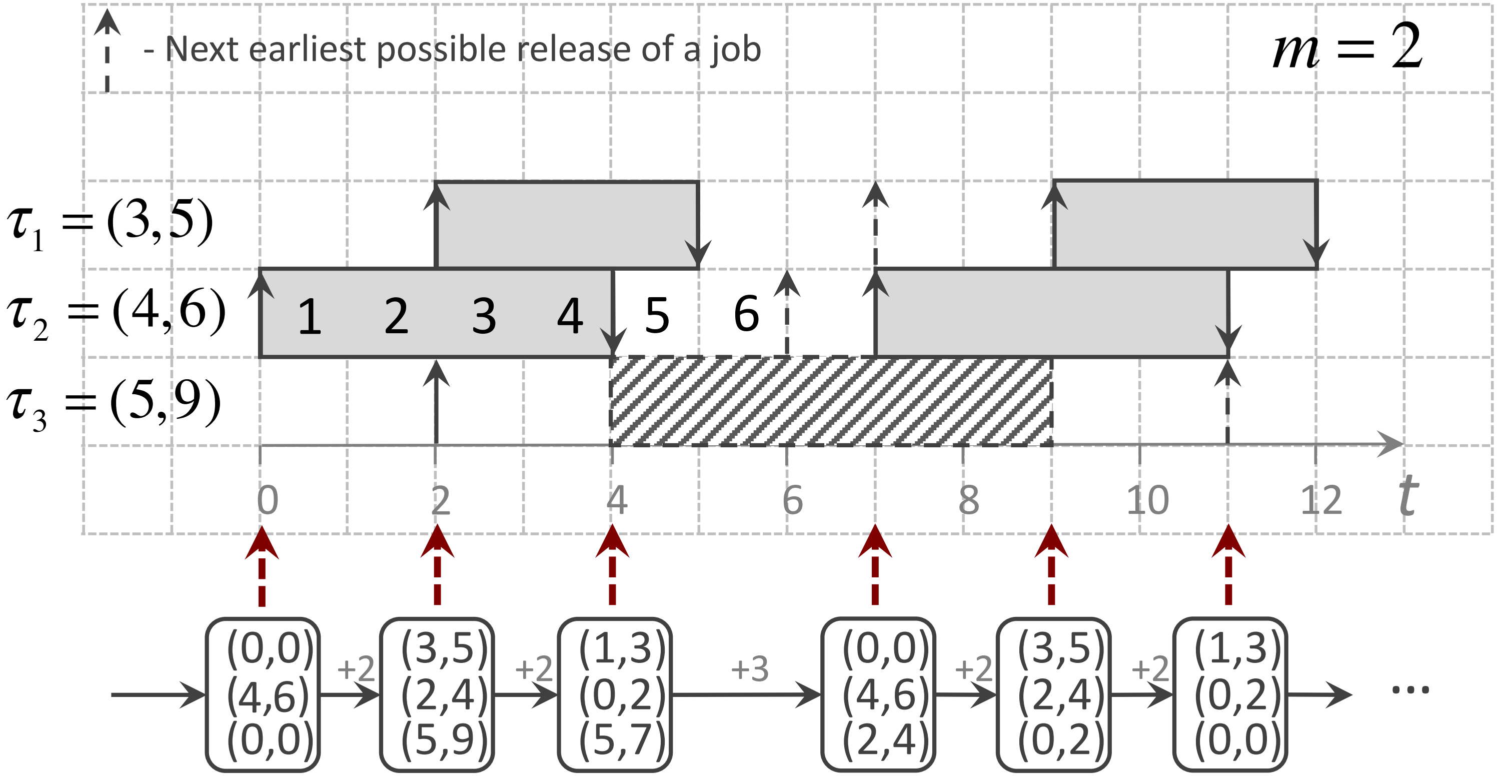
# State Transition Graph



# State Transition Graph

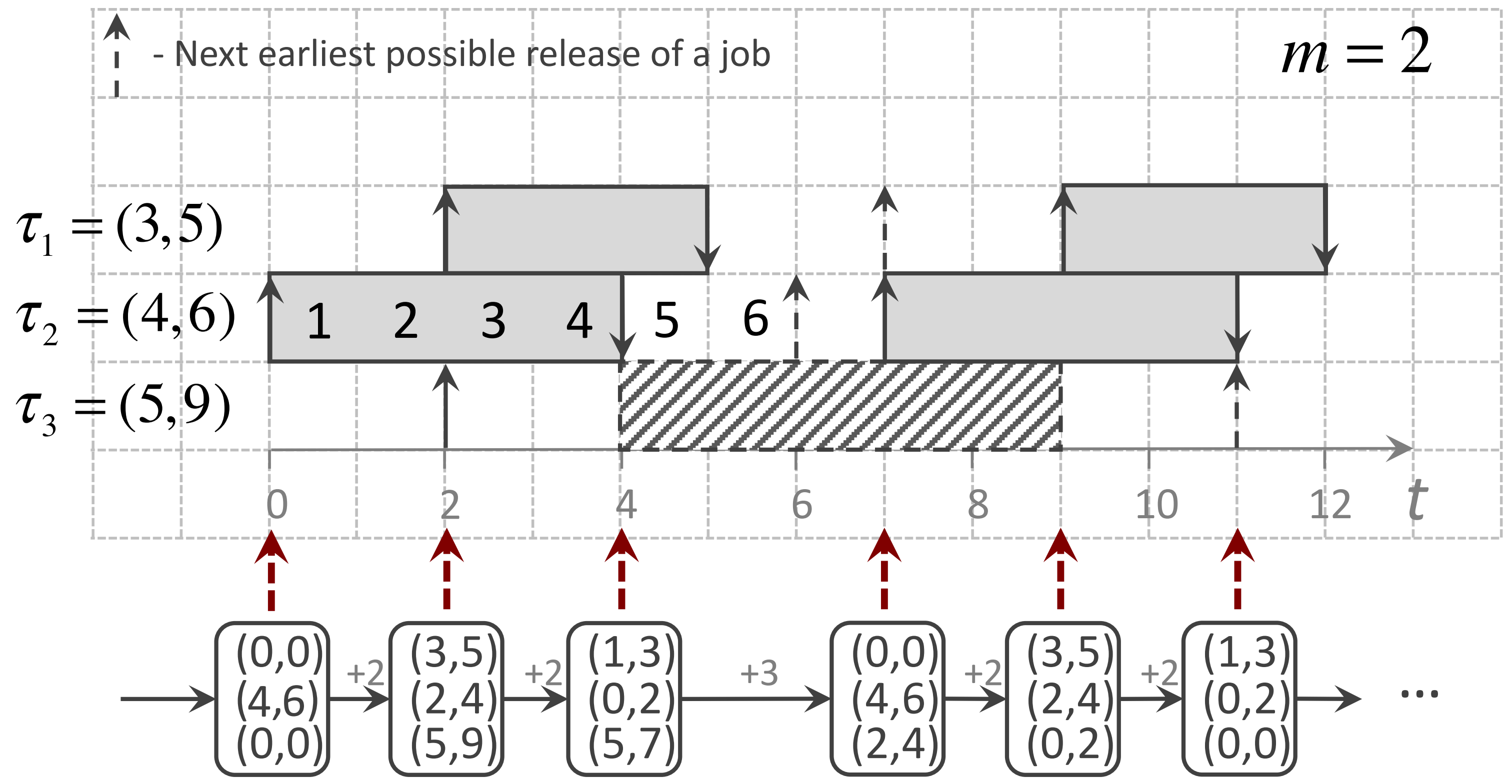


## State Transition Graph




Note: The release scenario above is just one out of many legal release scenarios

# State Transition Graph




Note: The release scenario above is just one out of many legal release scenarios

State transition graph presented on the next slides aims at modeling all possible schedules

# State Transition Graph Example

$\tau_1 = (2,5)$

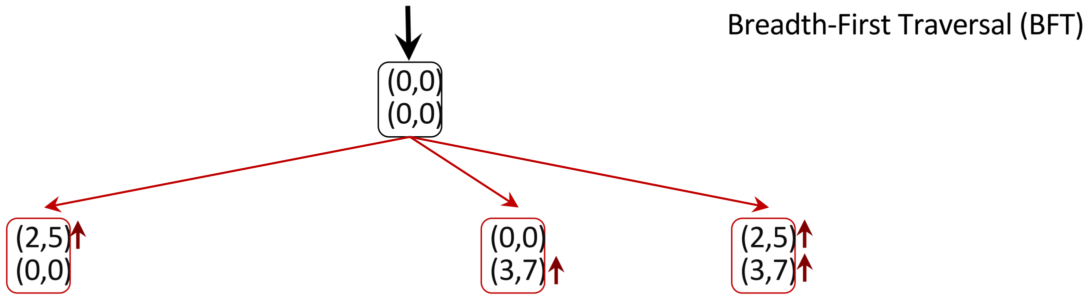
$\tau_2 = (3,7)$

$m = 1$

(0,0)
(0,0)

# State Transition Graph Example

$\tau_1 = (2,5)$
$\tau_2 = (3,7)$
$m = 1$

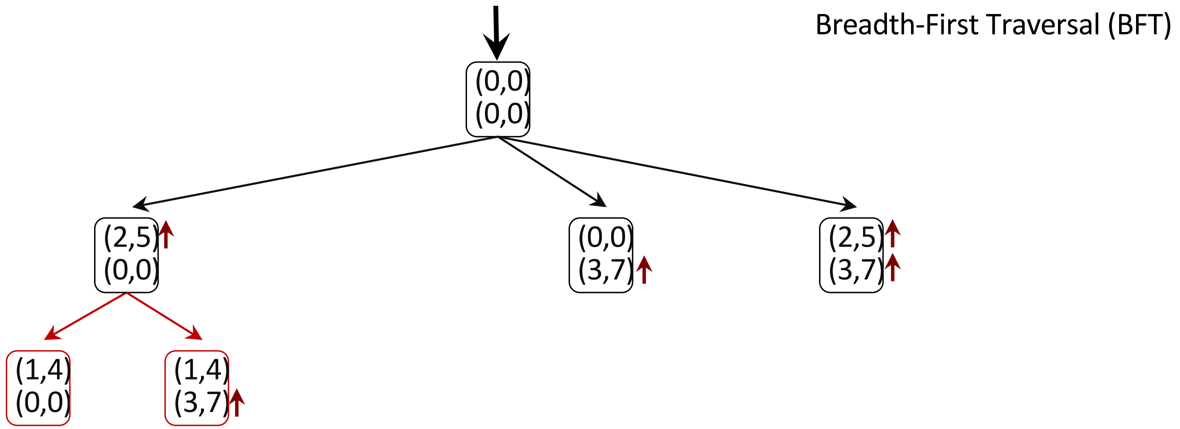


# State Transition Graph Example

$\tau_1 = (2,5)$

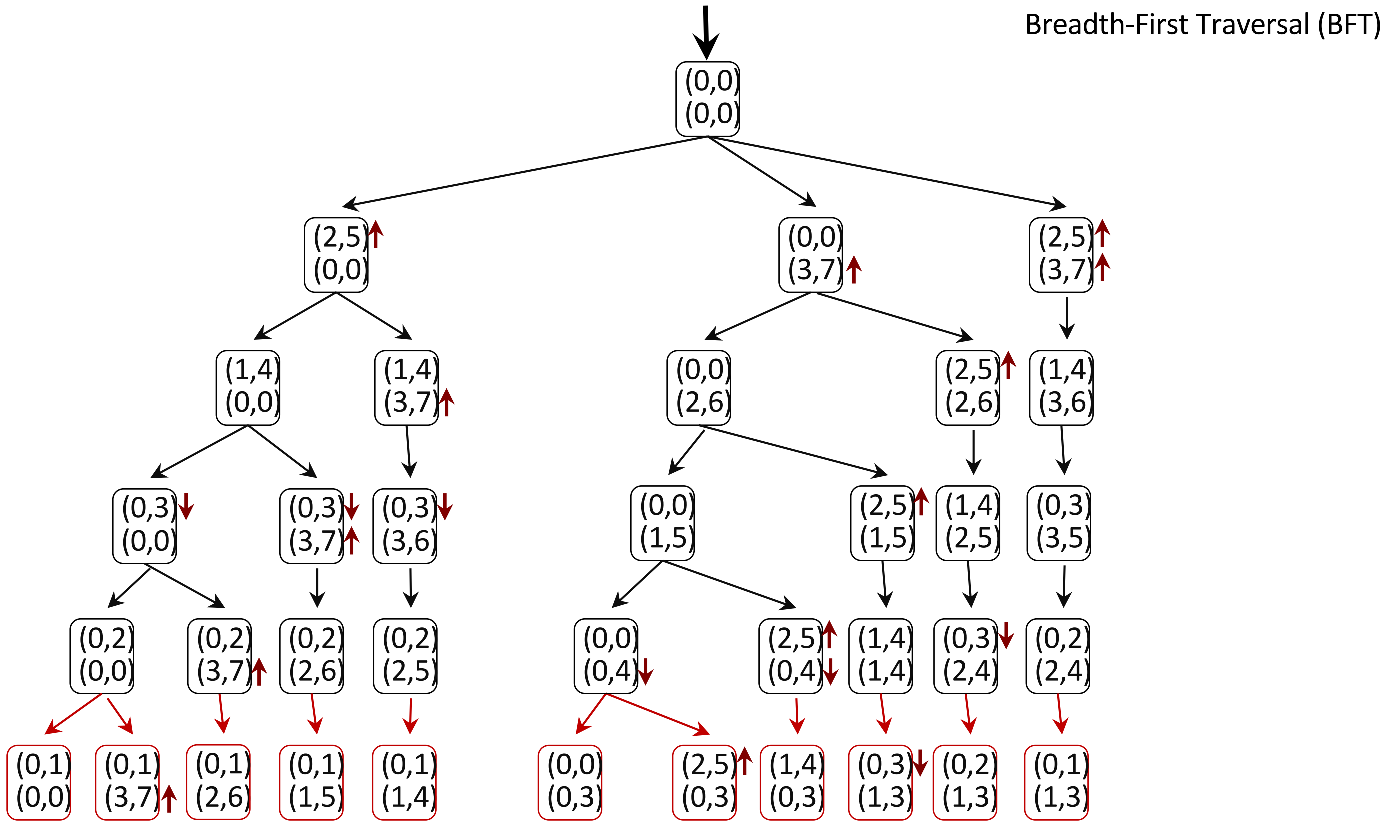
$\tau_2 = (3,7)$

$m = 1$



# State Transition Graph Example

$\tau_1 = (2,5)$
$\tau_2 = (3,7)$
$m = 1$

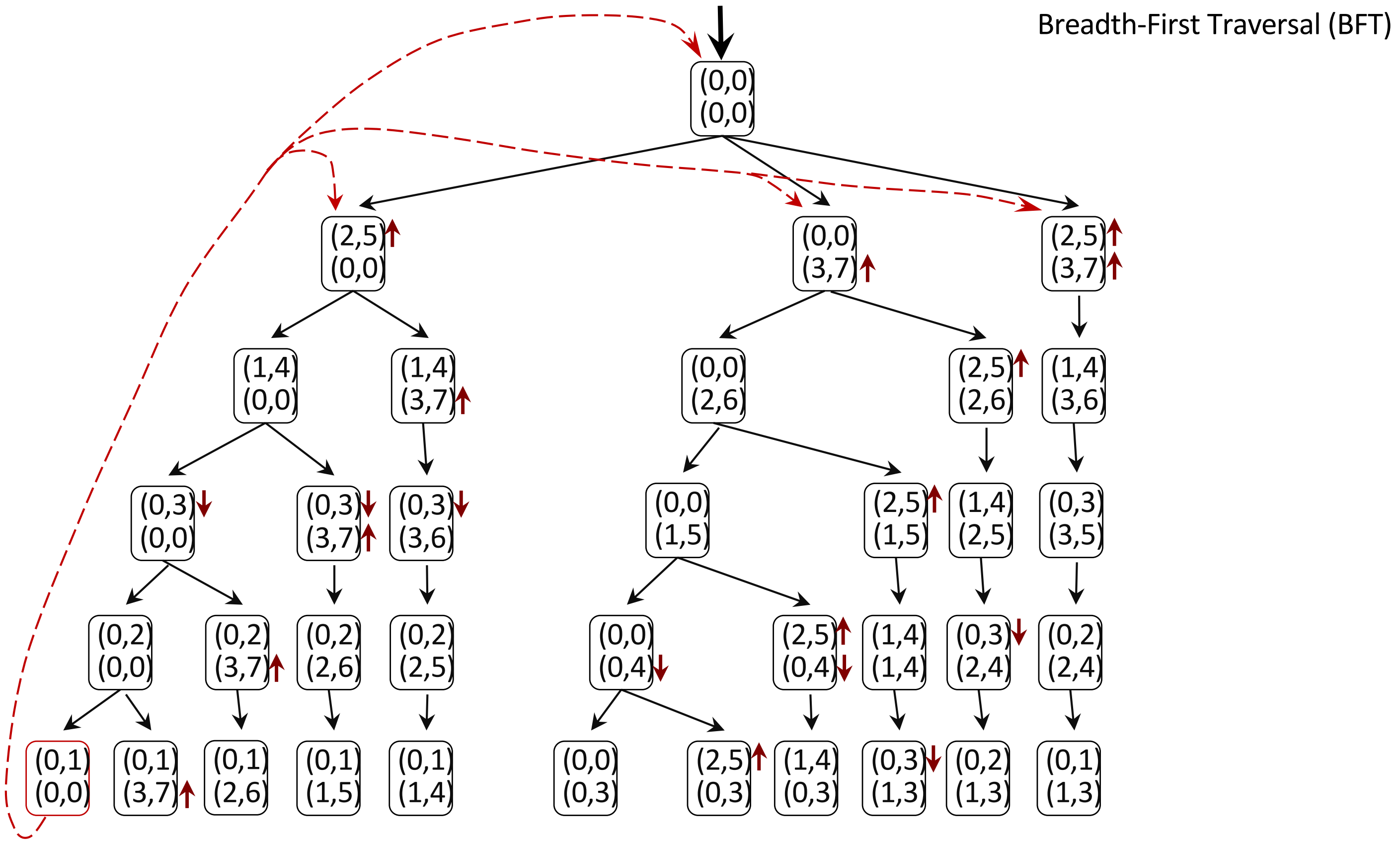


# State Transition Graph Example

$\tau_1 = (2,5)$

$\tau_2 = (3,7)$

$m = 1$

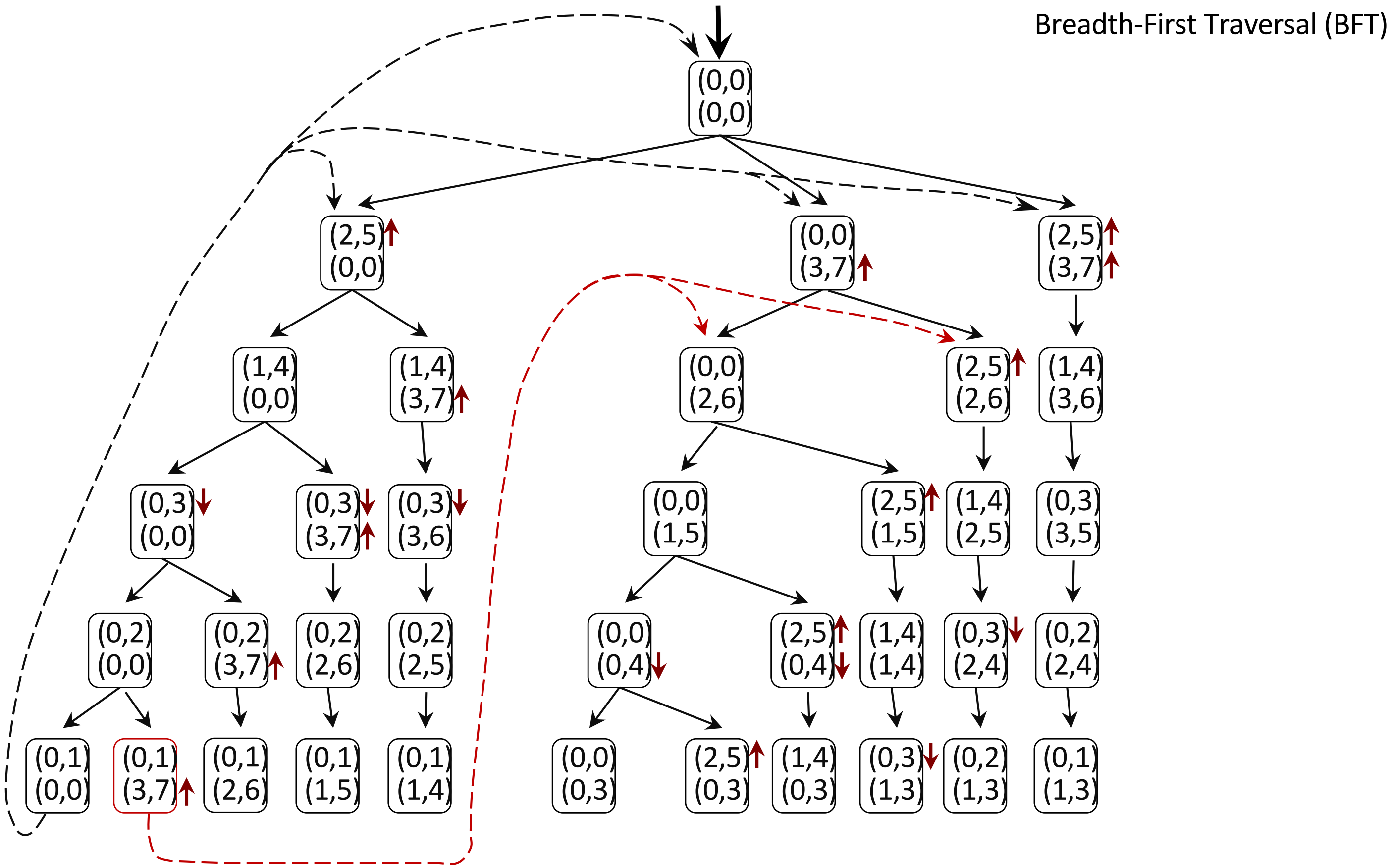


# State Transition Graph Example

$\tau_1 = (2,5)$

$\tau_2 = (3,7)$

$m = 1$

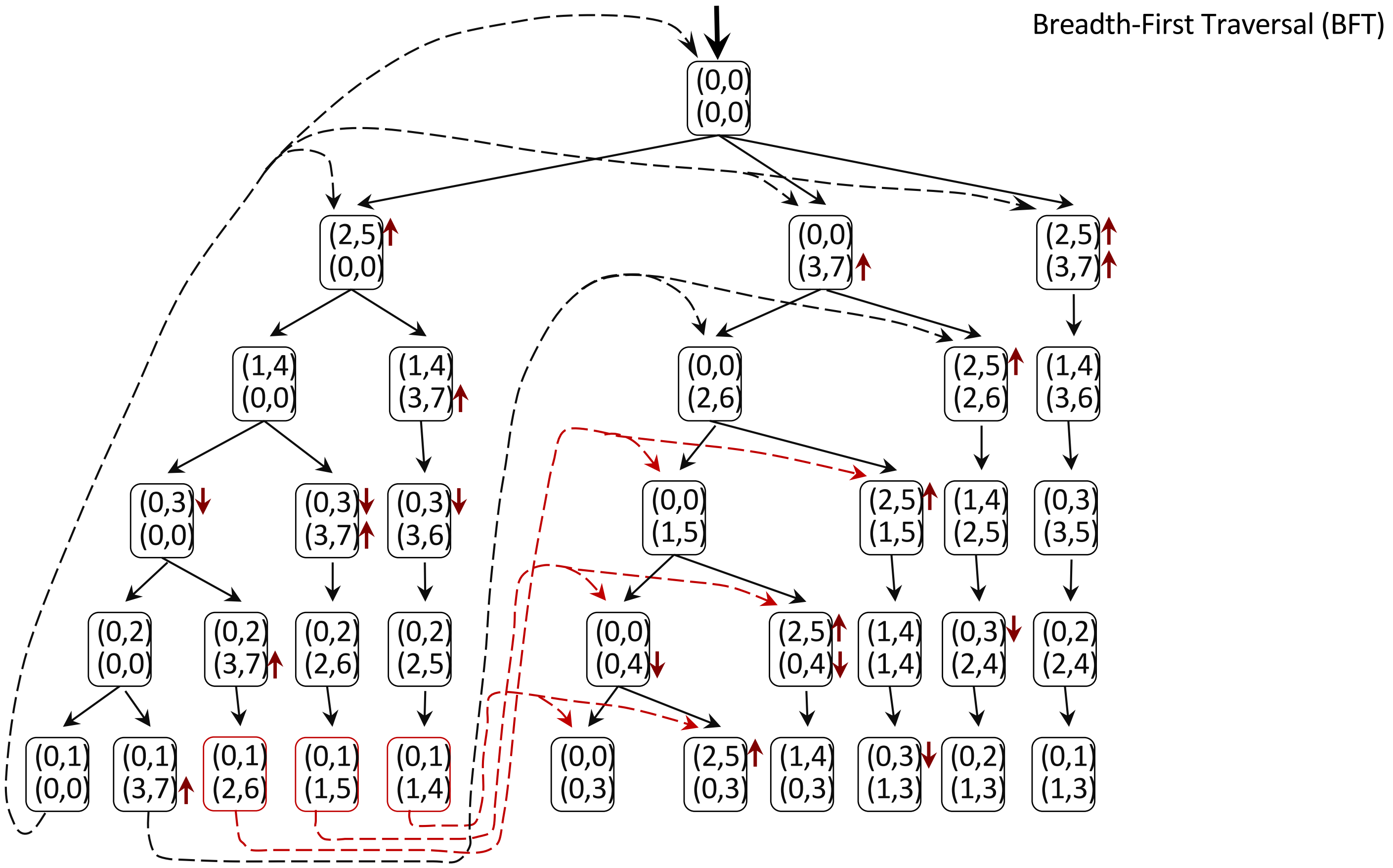


# State Transition Graph Example

$\tau_1 = (2,5)$

$\tau_2 = (3,7)$

$m = 1$

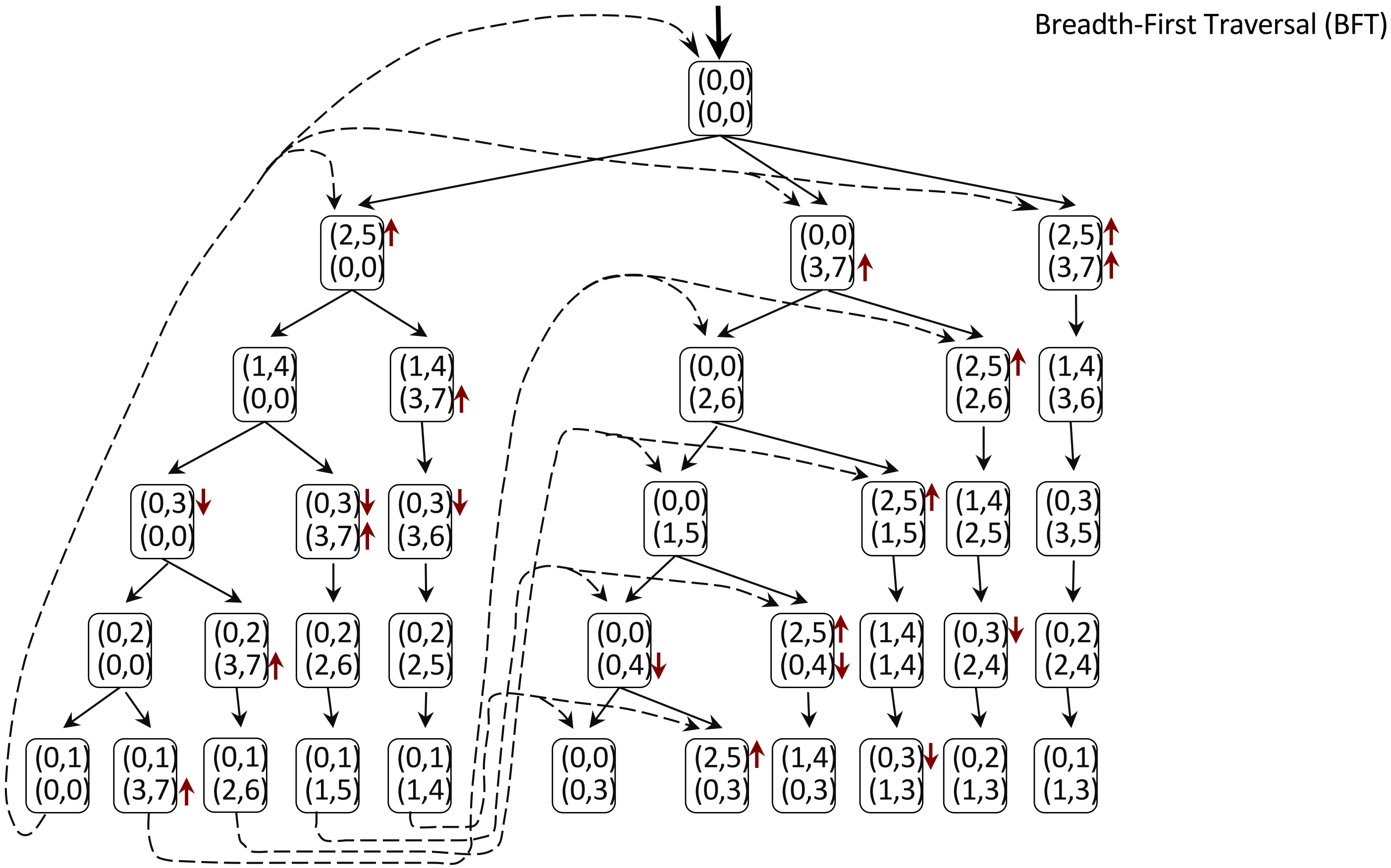


# State Transition Graph Example

$\tau_1 = (2,5)$

$\tau_2 = (3,7)$

$m = 1$

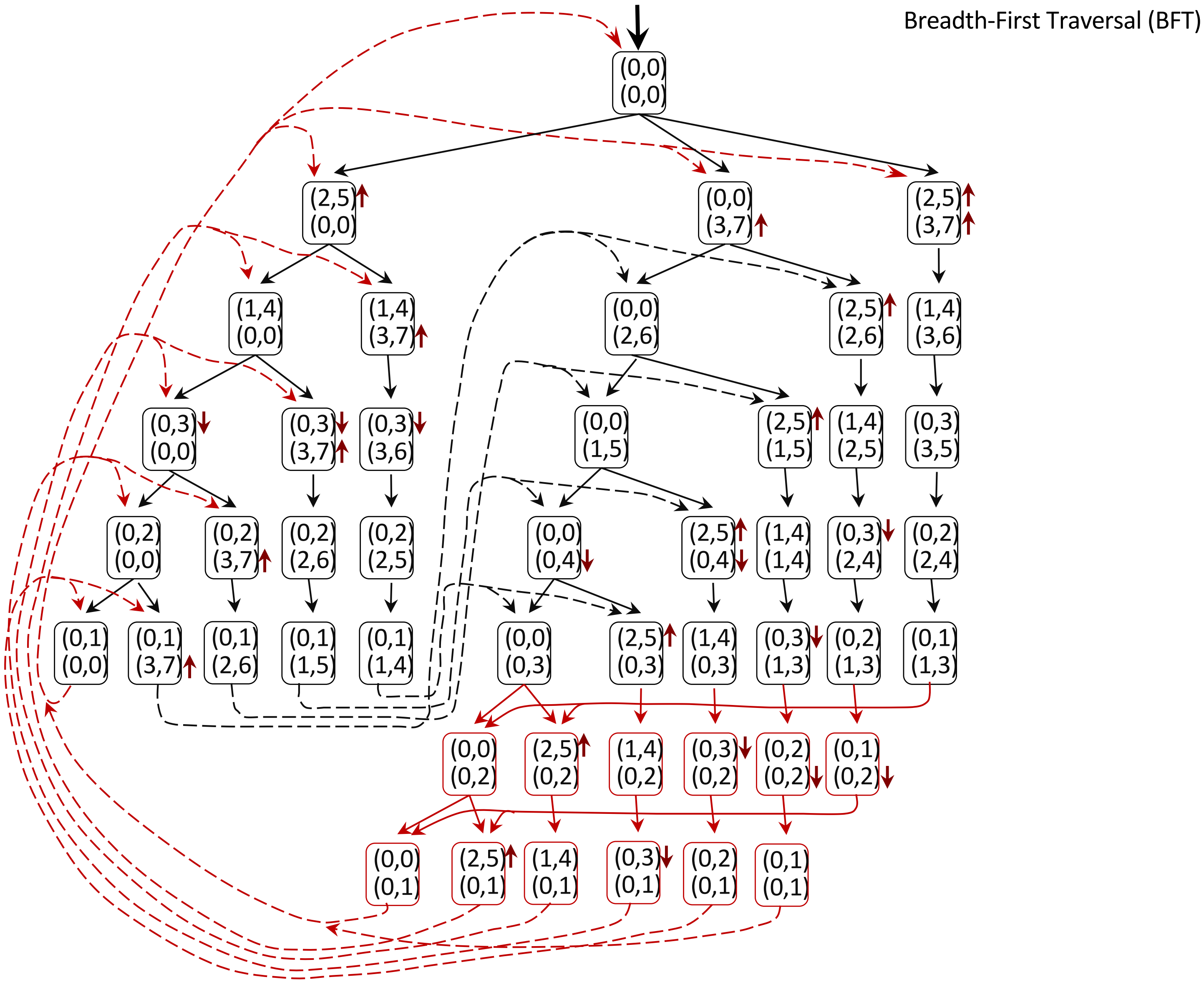


## State Transition Graph Example

$\tau_1 = (2,5)$

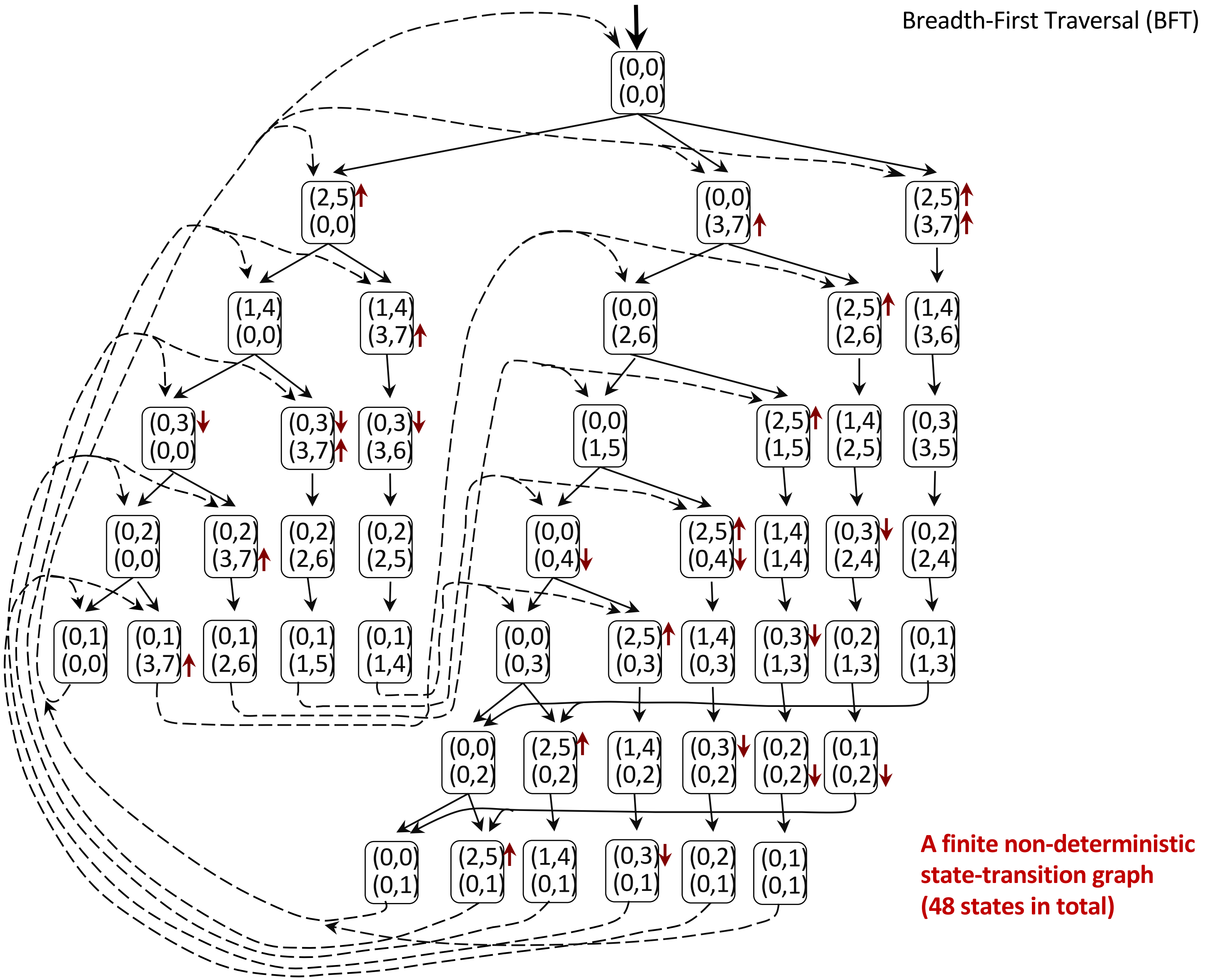
$\tau_2 = (3,7)$

$m = 1$



## State Transition Graph Example

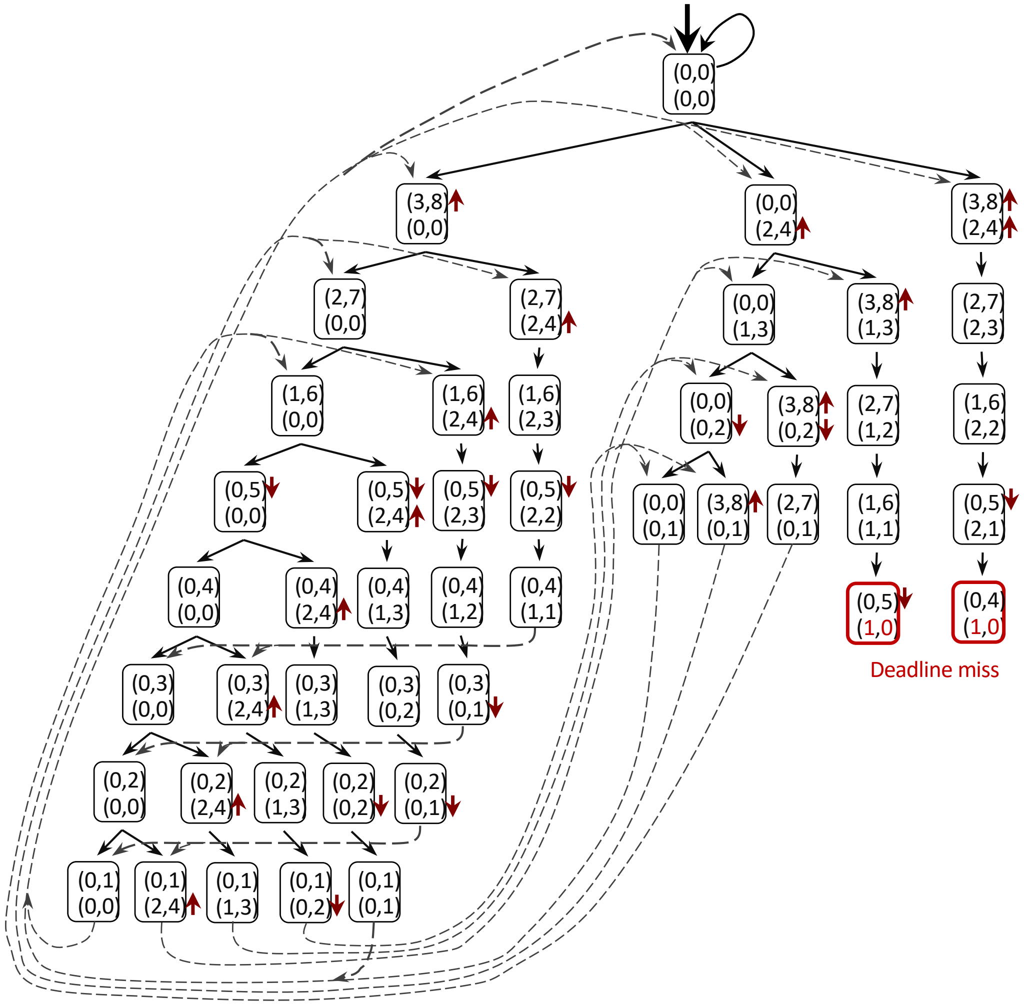
$\tau_1 = (2,5)$
$\tau_2 = (3,7)$
$m = 1$




**A finite non-deterministic state-transition graph (48 states in total)**

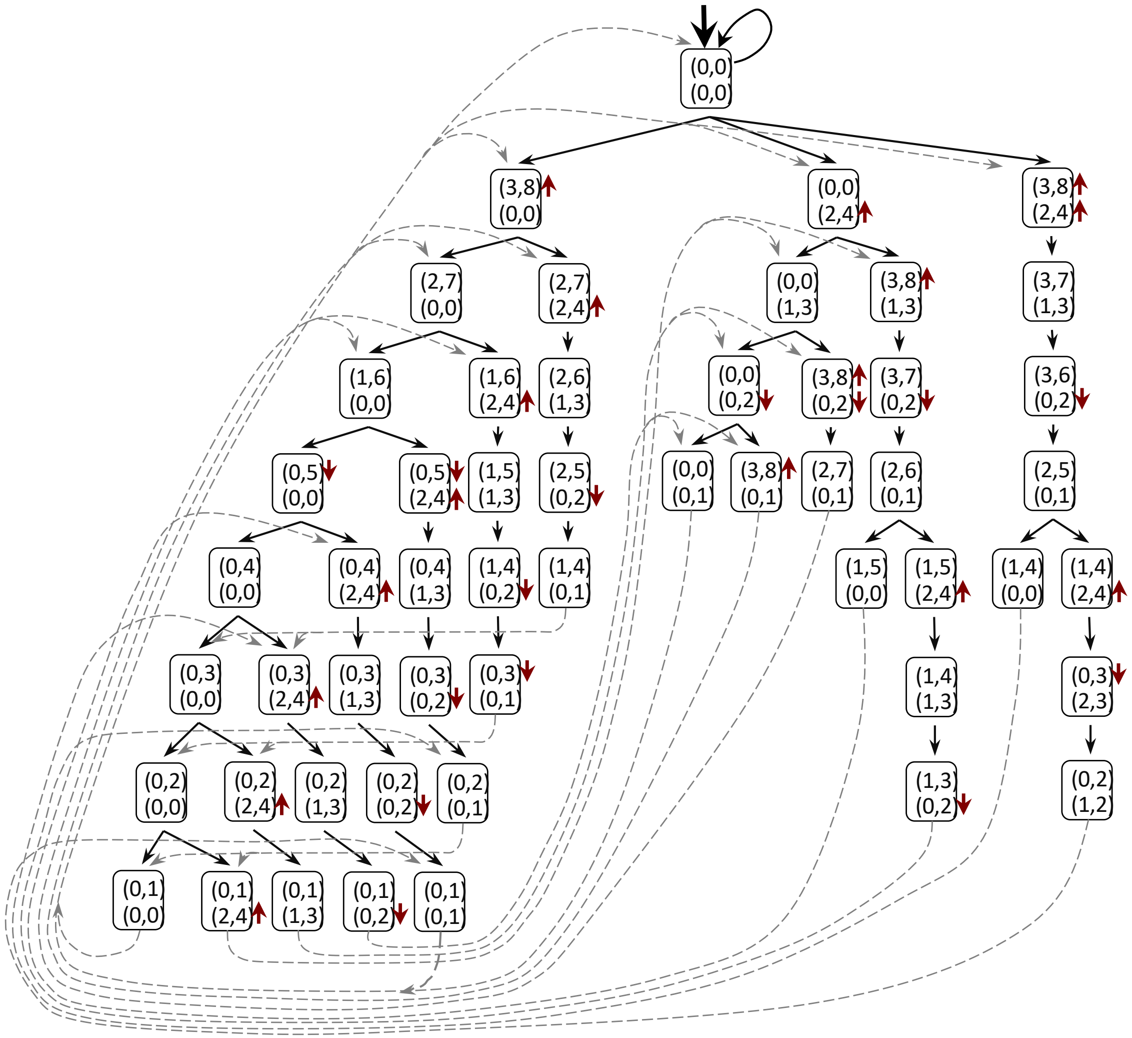
# Another Example for GFP with a Deadline Miss

$\tau_1 = (3,8)$
$\tau_2 = (2,4)$
$m = 1$



# Another Example for GEDF with NO Deadline Miss

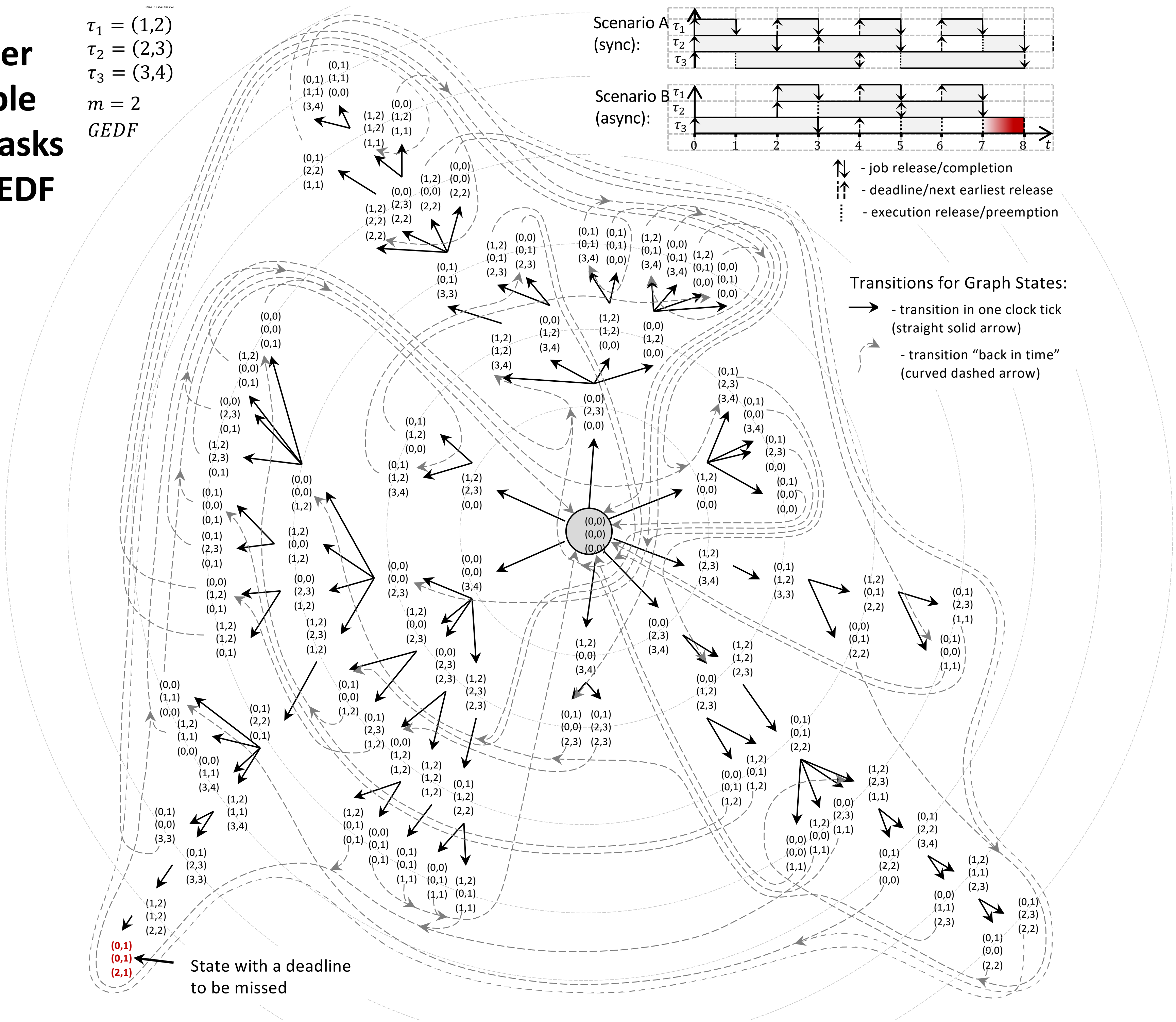
$\tau_1 = (3{,}8)$
$\tau_2 = (2{,}4)$
$\mathrm{m} = 1$




No deadline miss!
(unlike FP scheduling on the previous slide)

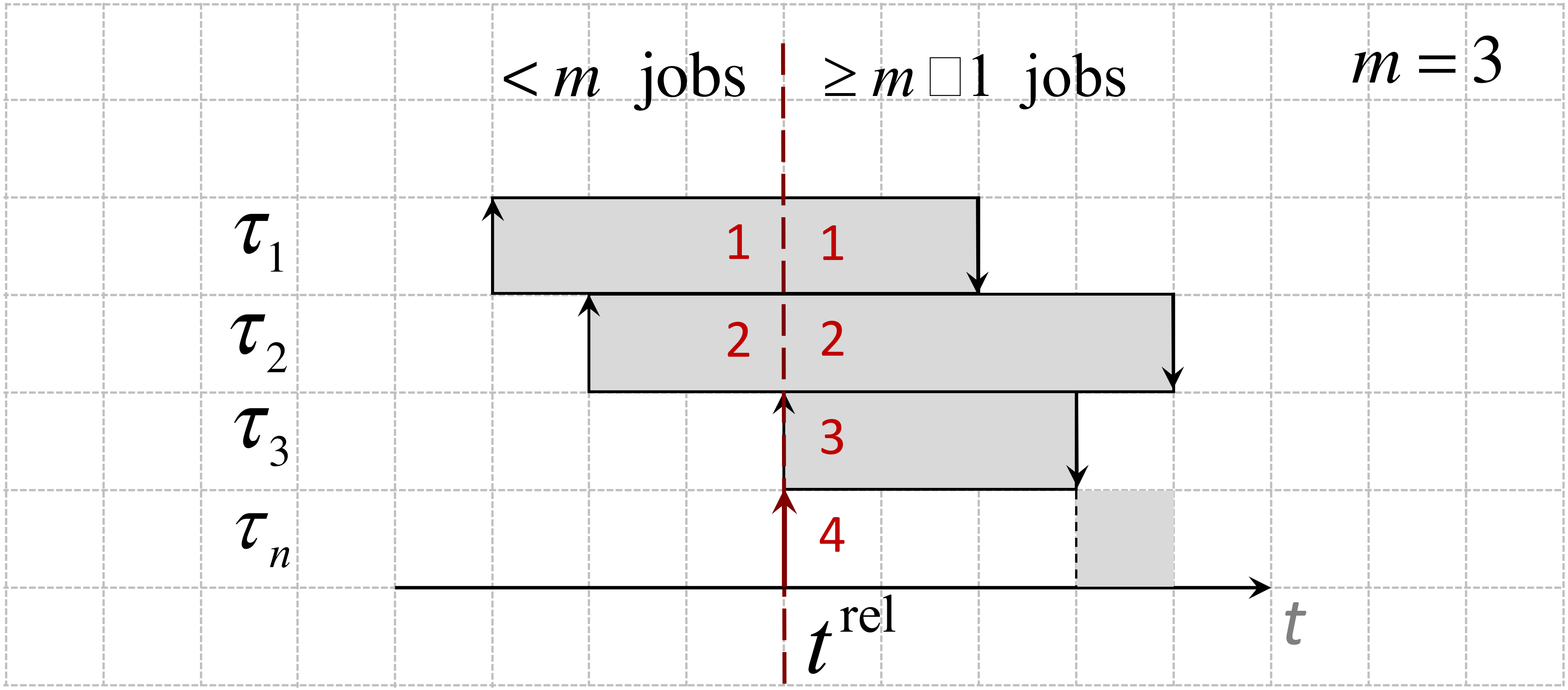
# Another Example for 3 tasks and GEDF

$\tau_1 = (1,2)$
$\tau_2 = (2,3)$
$\tau_3 = (3,4)$
$m = 2$
$GEDF$



# Pruning:
# Critical Release Instant of Davis and Burns (2011):

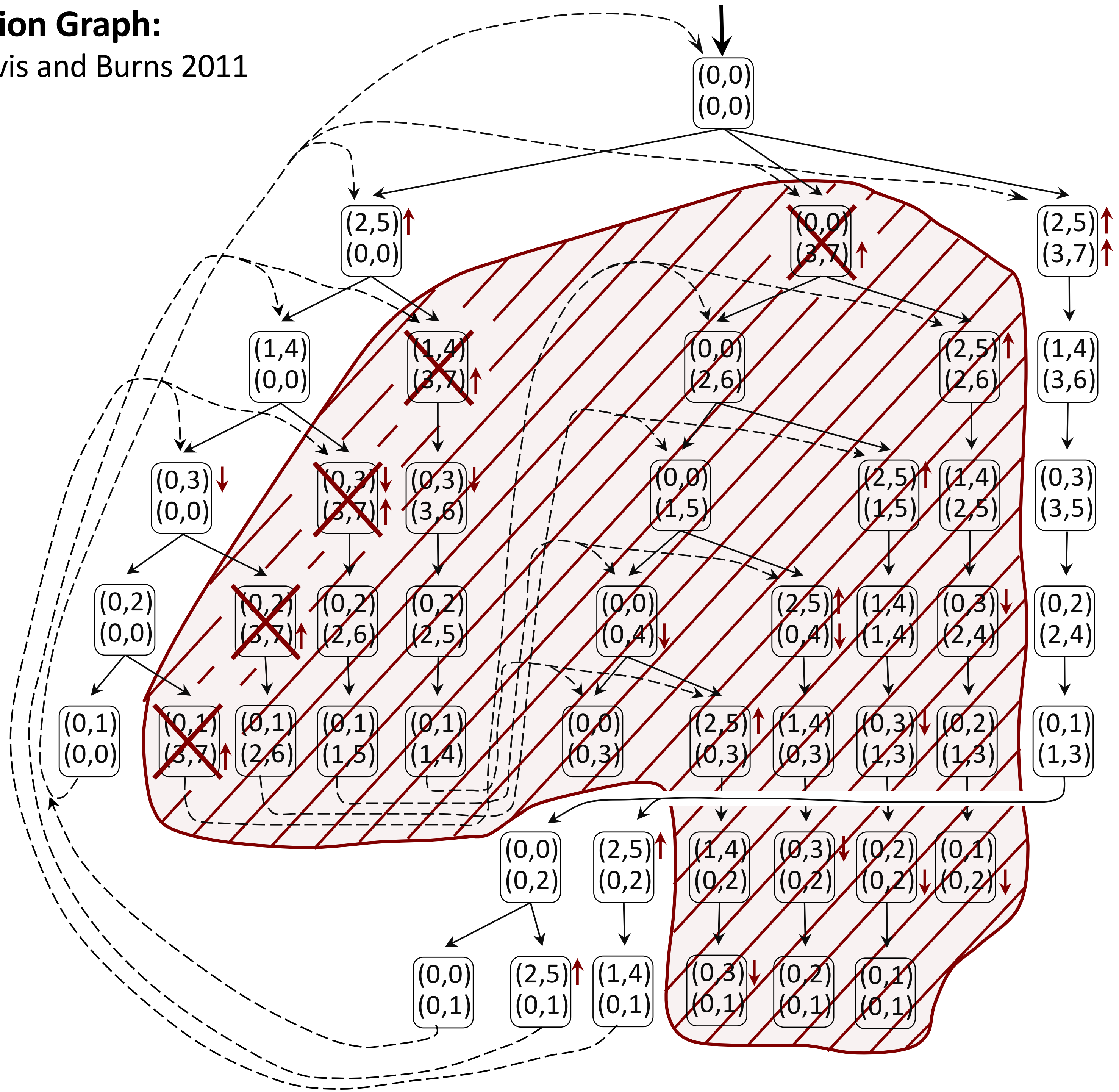
Sufficient to consider only such releases $t^{\text{rel}}$ for $\tau_n$, that satisfy the following constraints



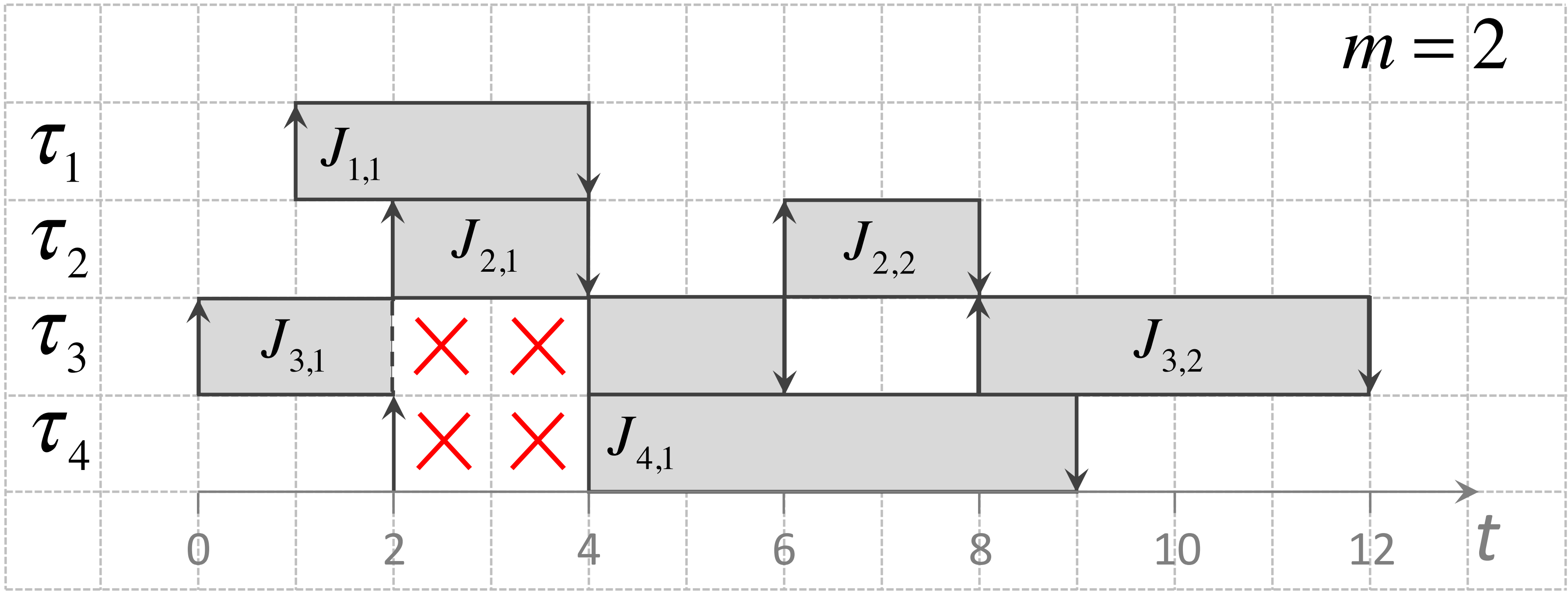
## State Transition Graph:

Pruning by Davis and Burns 2011

$$\tau_1 = (2,5)$$
$$\tau_2 = (3,7)$$
$$m = 1$$

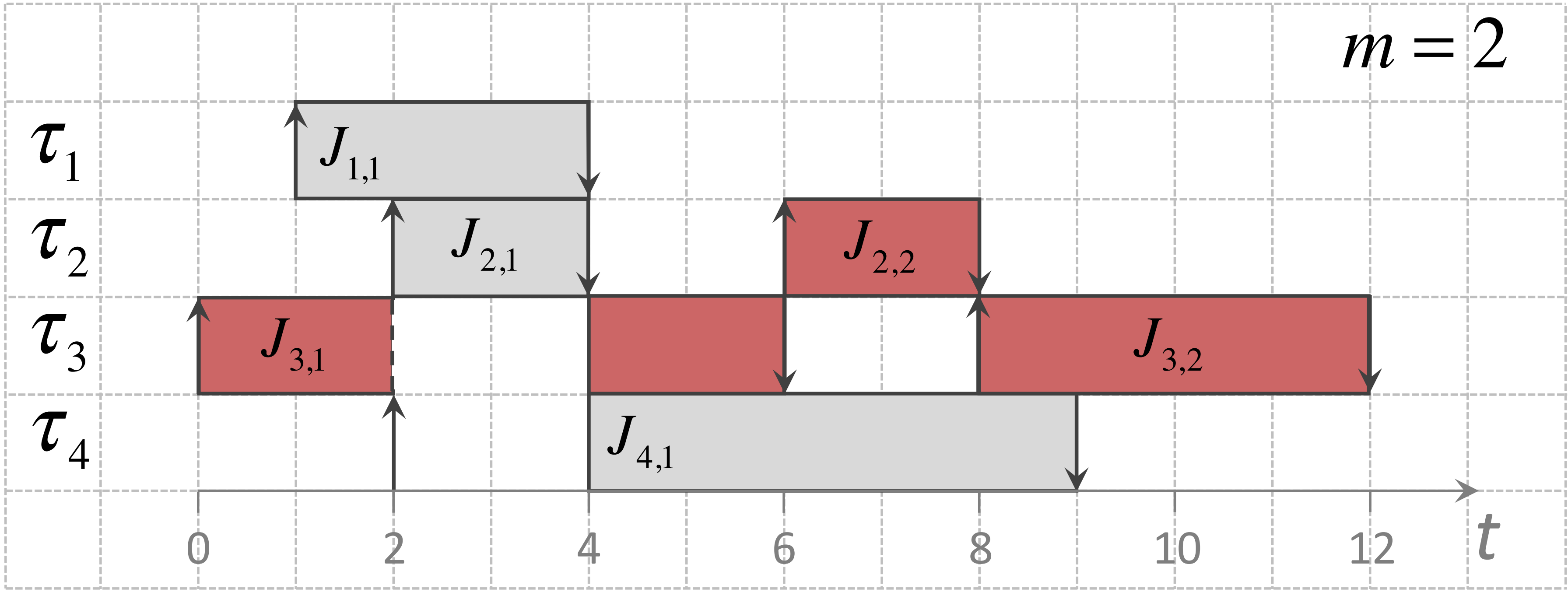


# Pruning Constraint 2: Job Interference



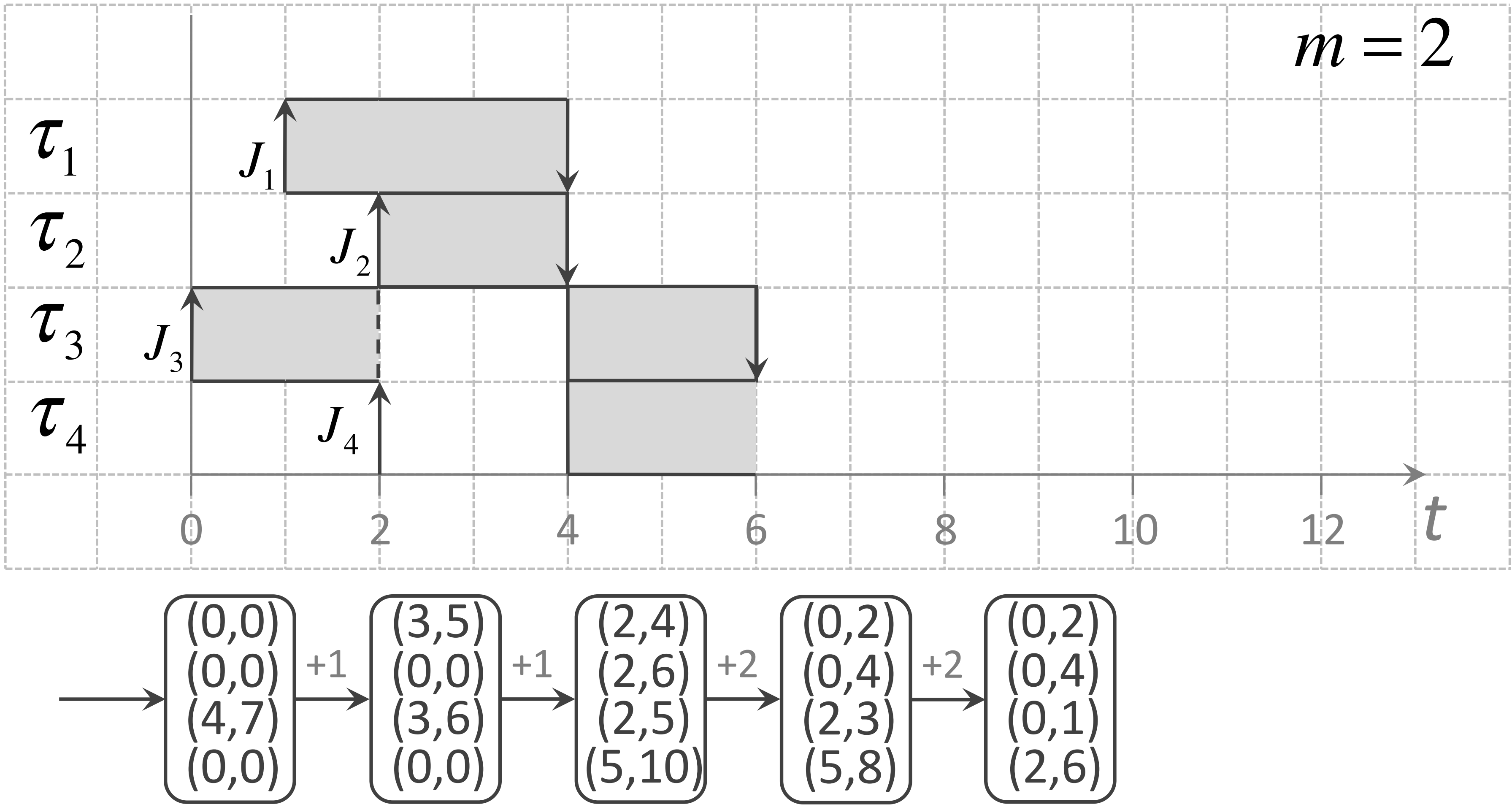

 - Job is pending, but no resource allocated

# Pruning Constraint 2: Job Interference

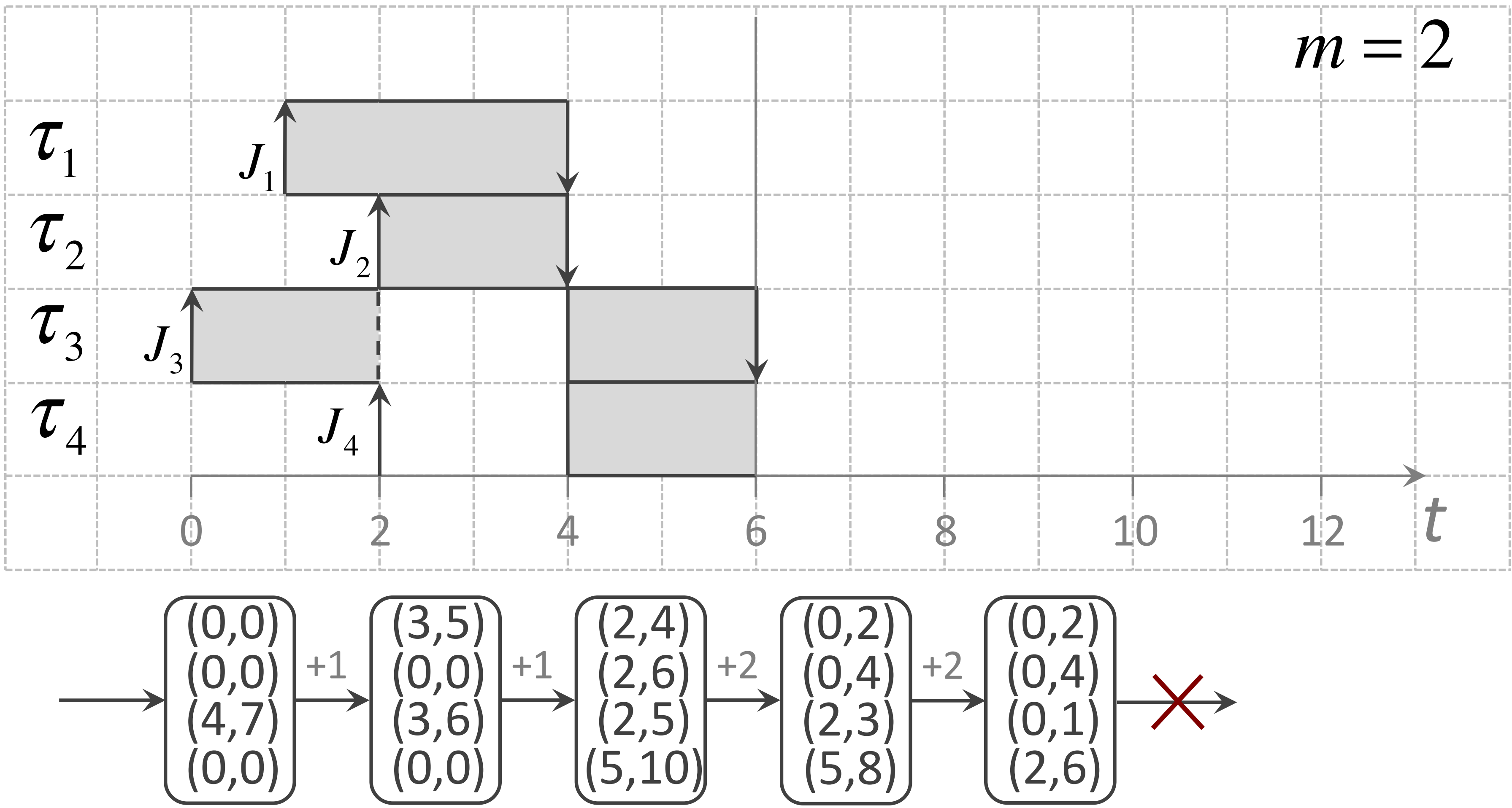



All jobs, not interfering with lower-priority ones, can be removed

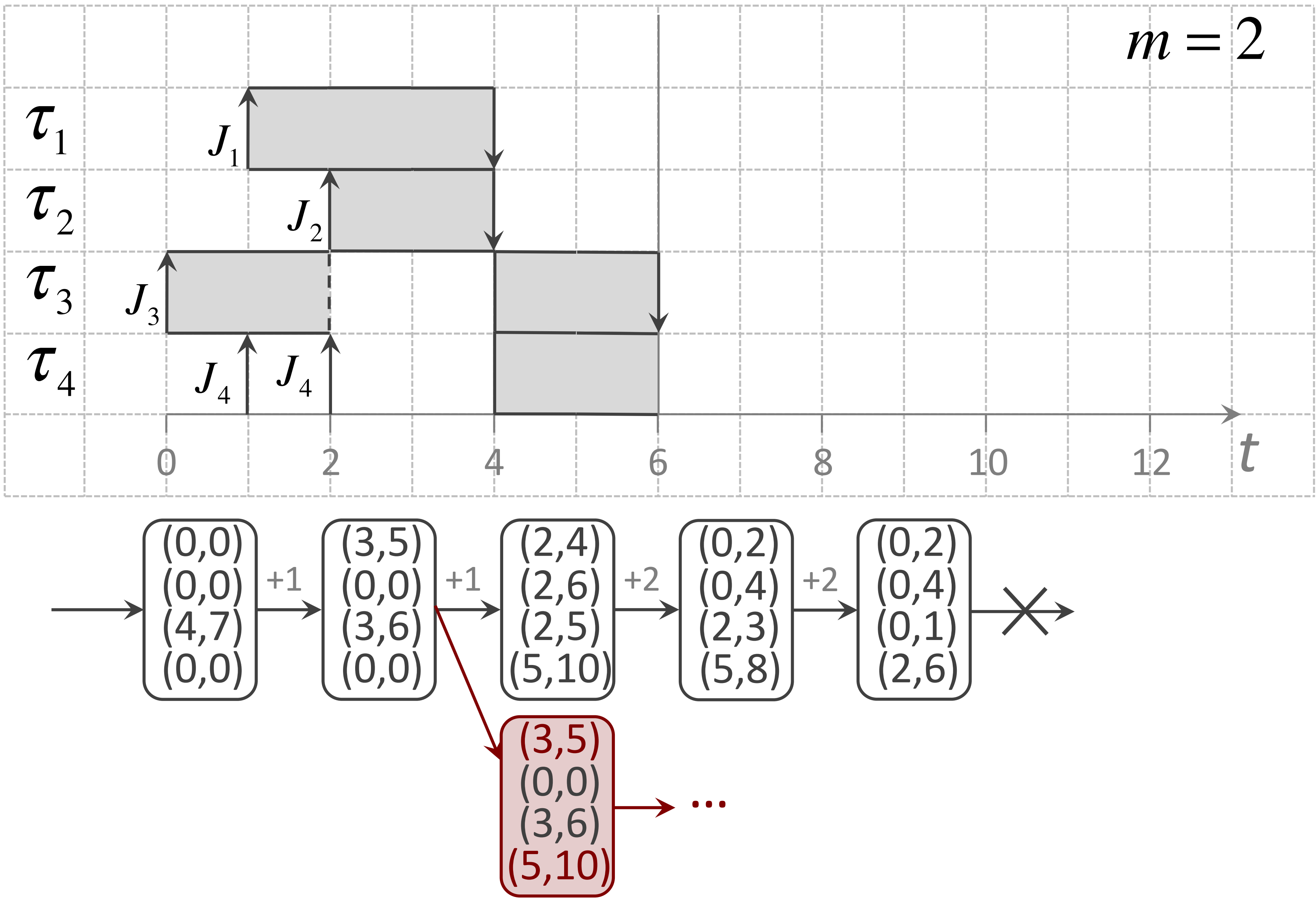
# Pruning Constraint 2:
# Job Interference



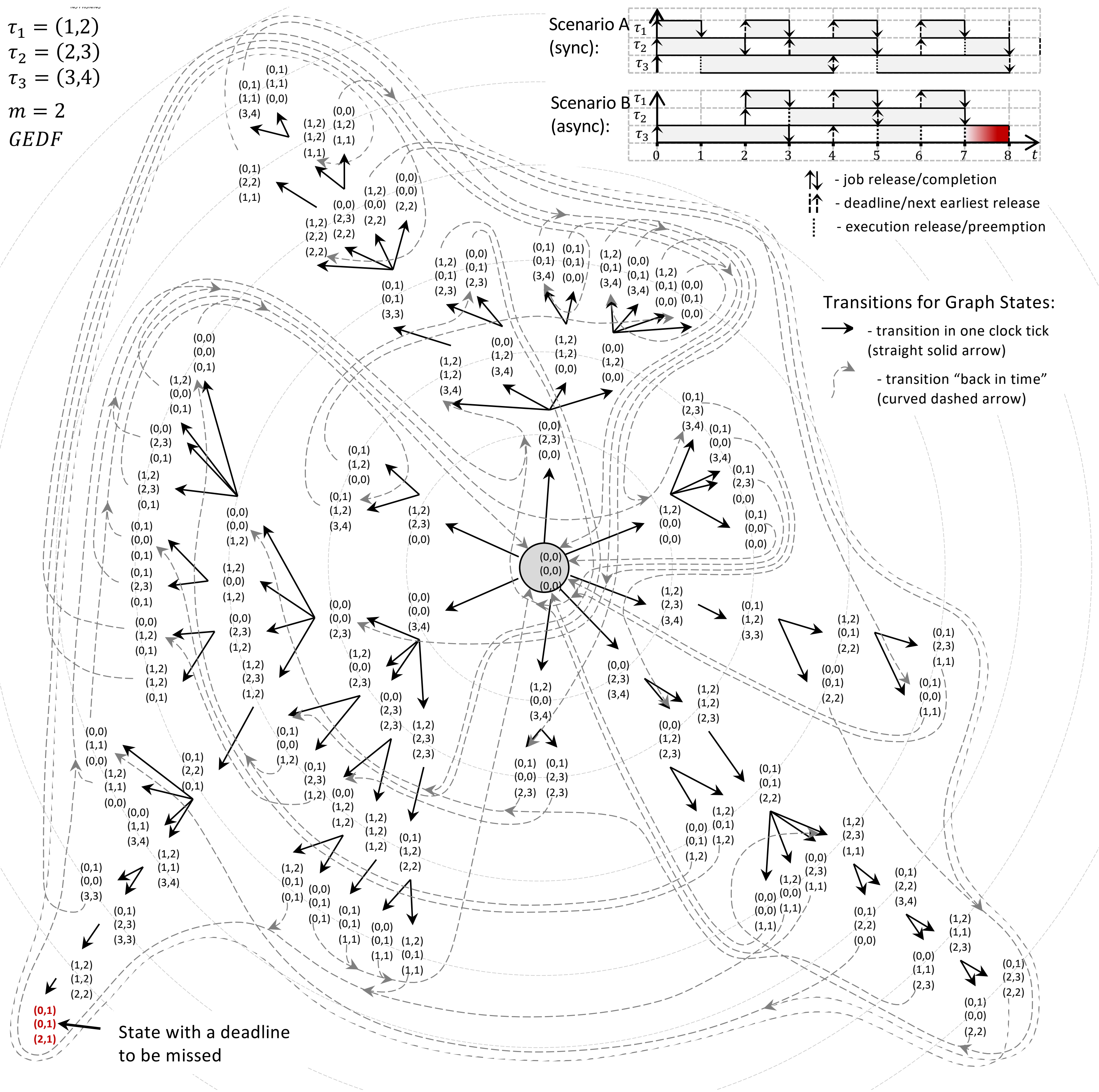
# Pruning Constraint 2: Job Interference



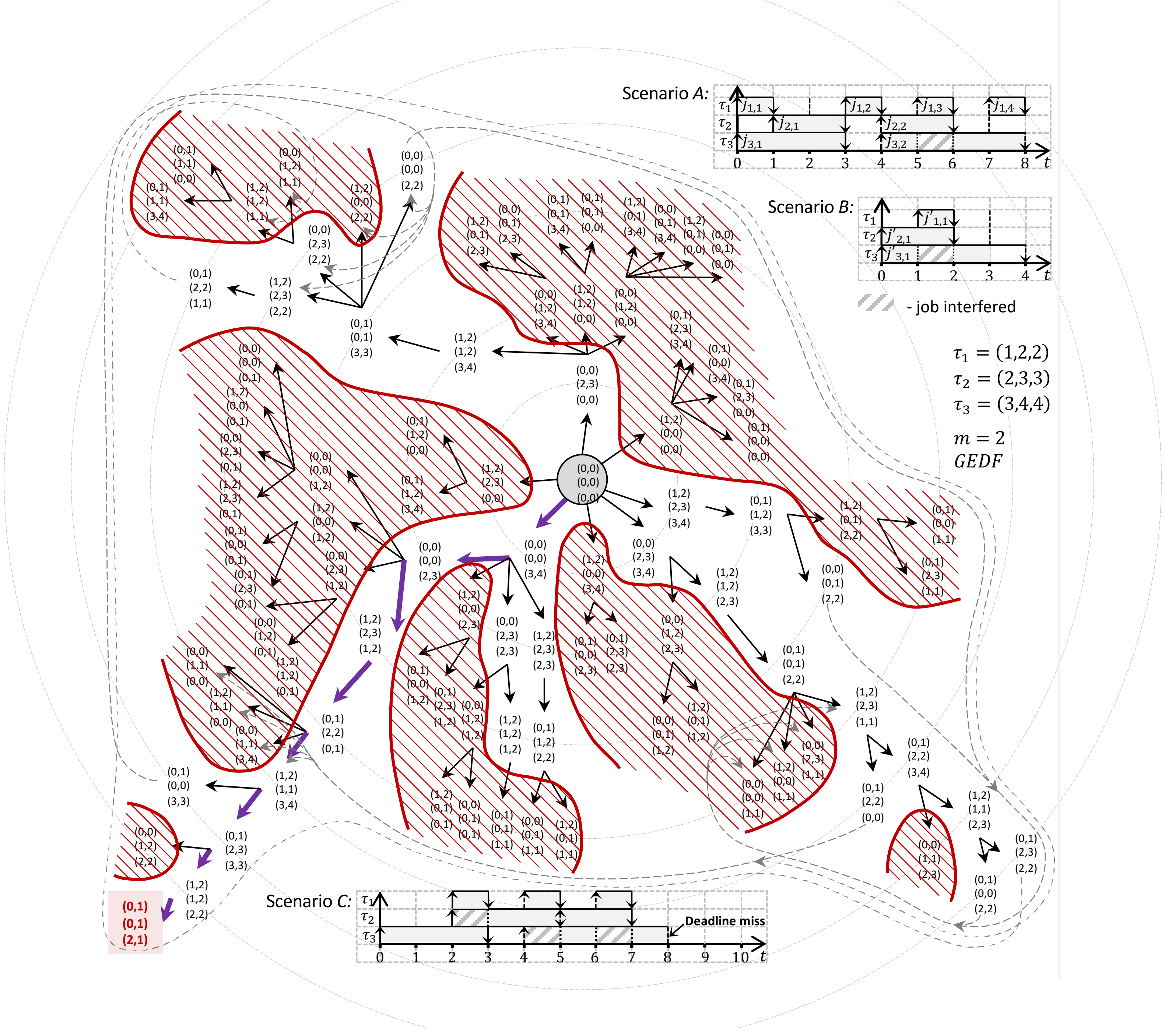
# Pruning Constraint 2:
# Job Interference



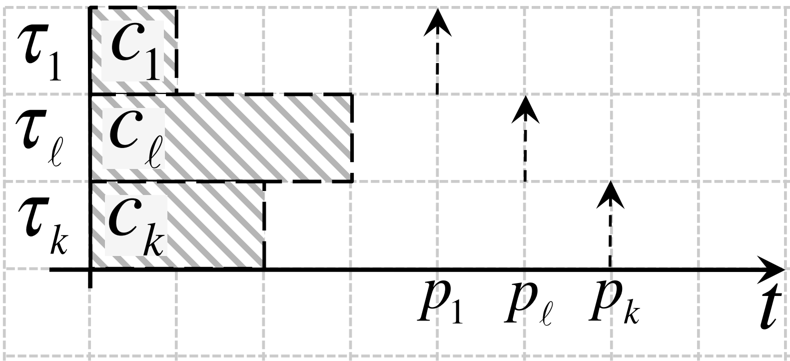
# Non-pruned graph

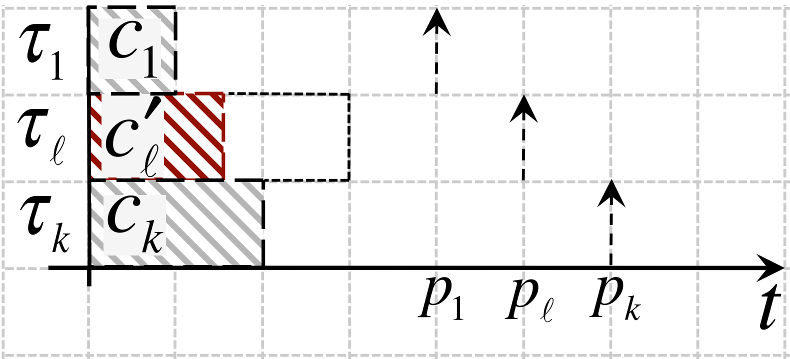


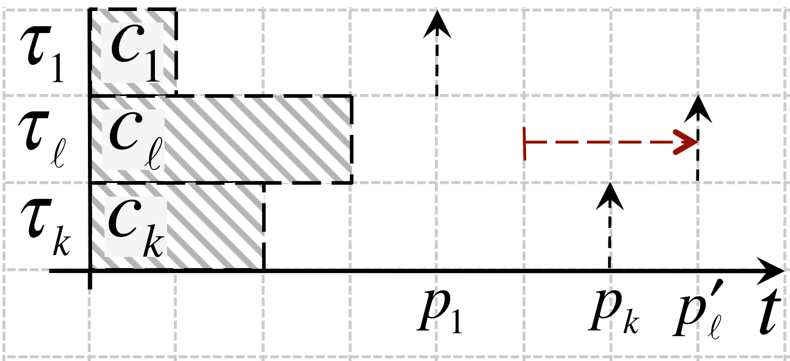
# Graph Pruned by a Job Interference Requirement



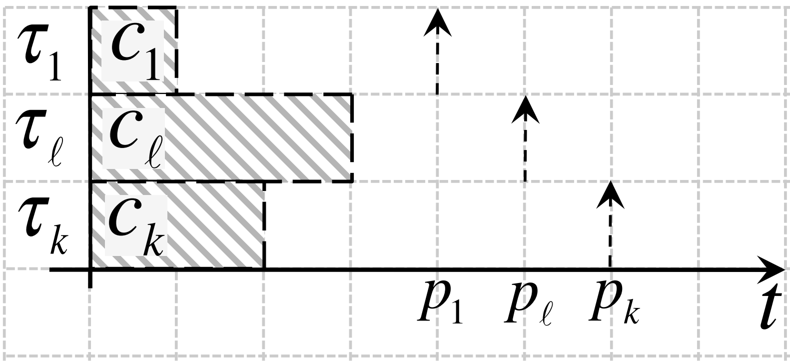
# Pruning Based on the Worst-Case States Notion



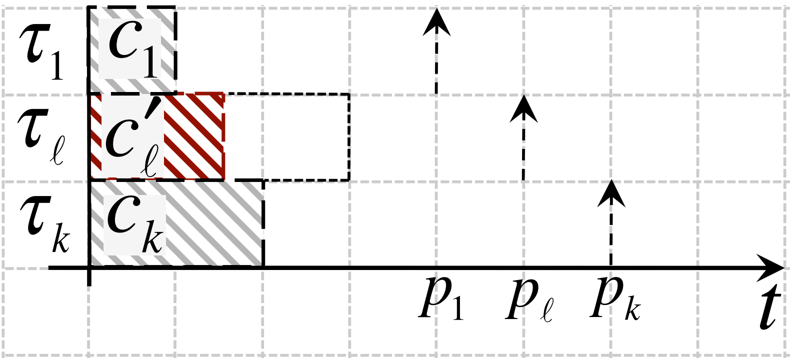



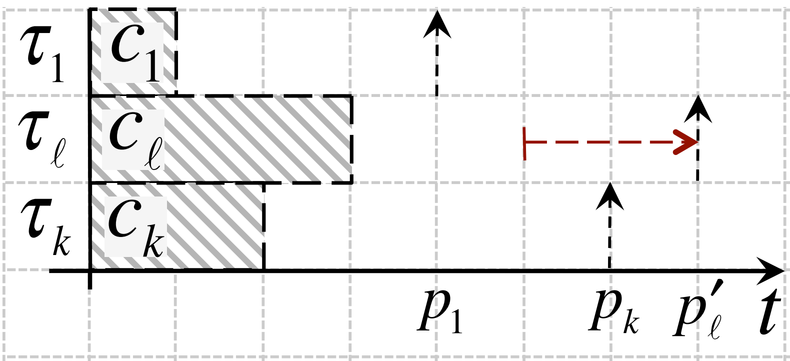



# Pruning Based on the Worst-Case States Notion

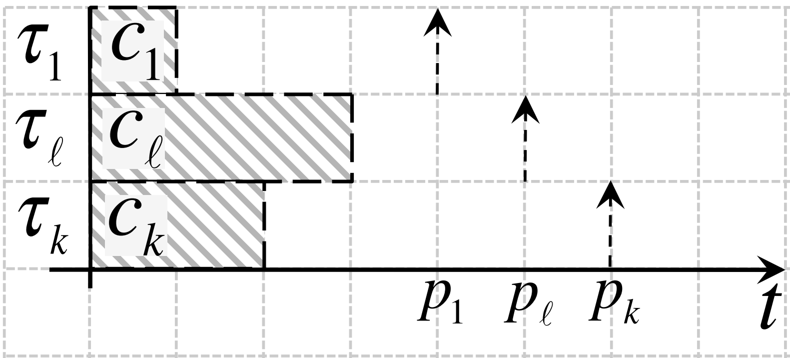


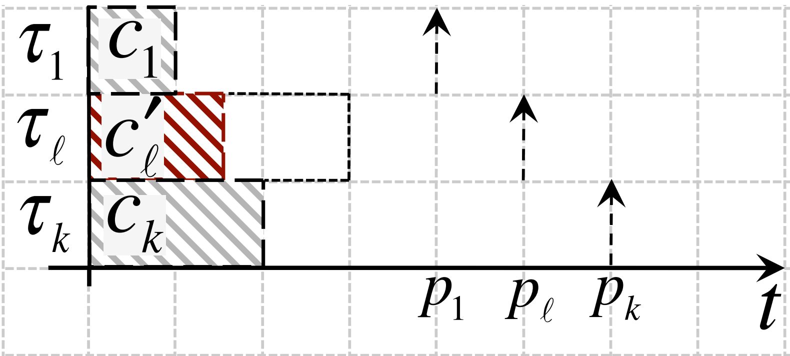

**<- This is the worst-case among these three scenarios**

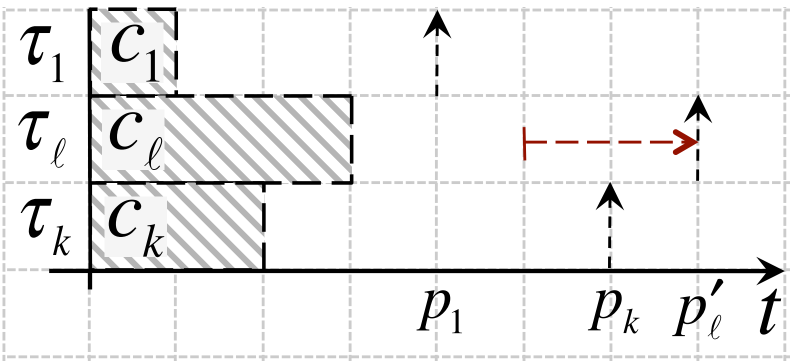


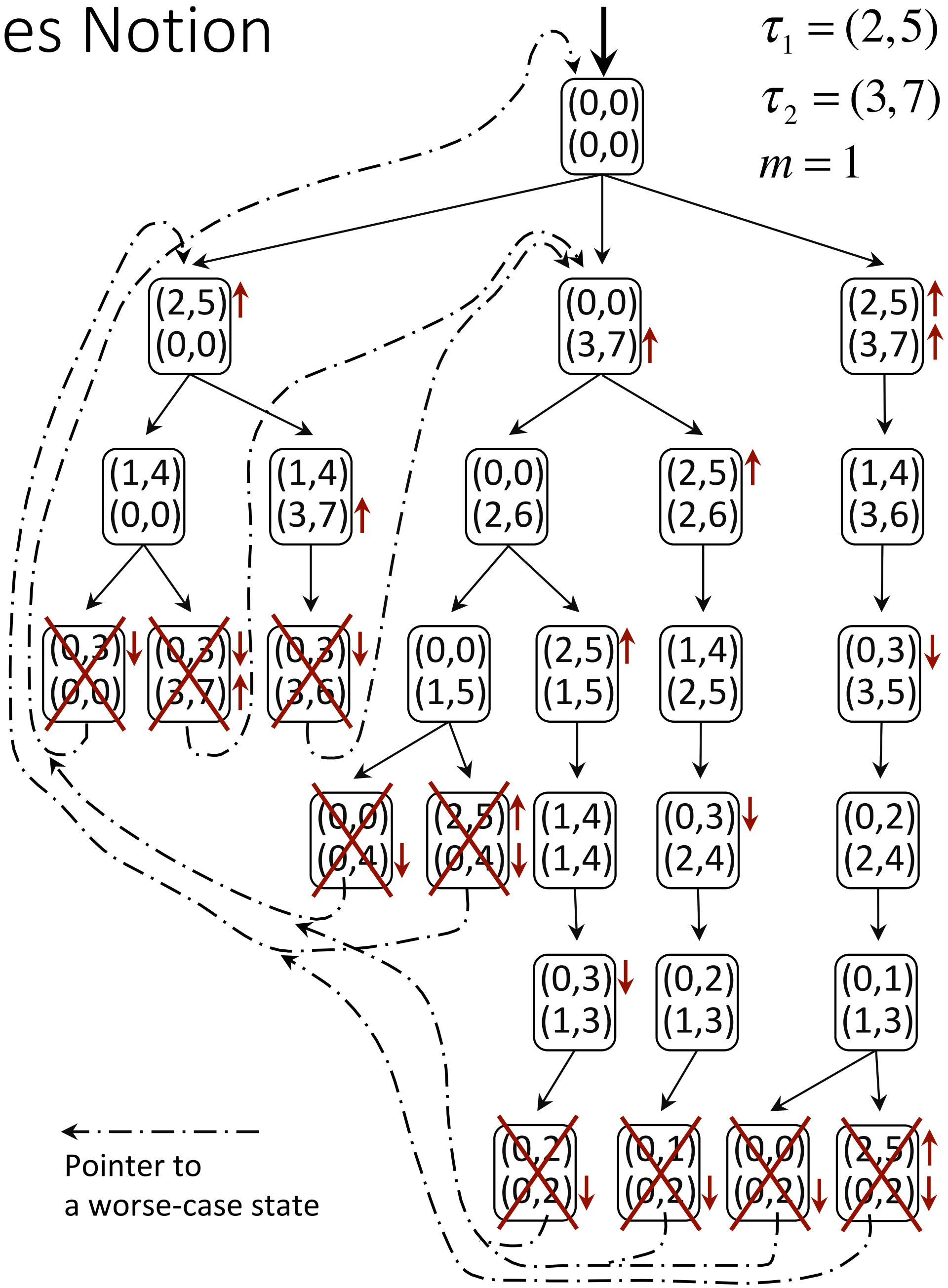



# Pruning Based on the Worst-Case States Notion

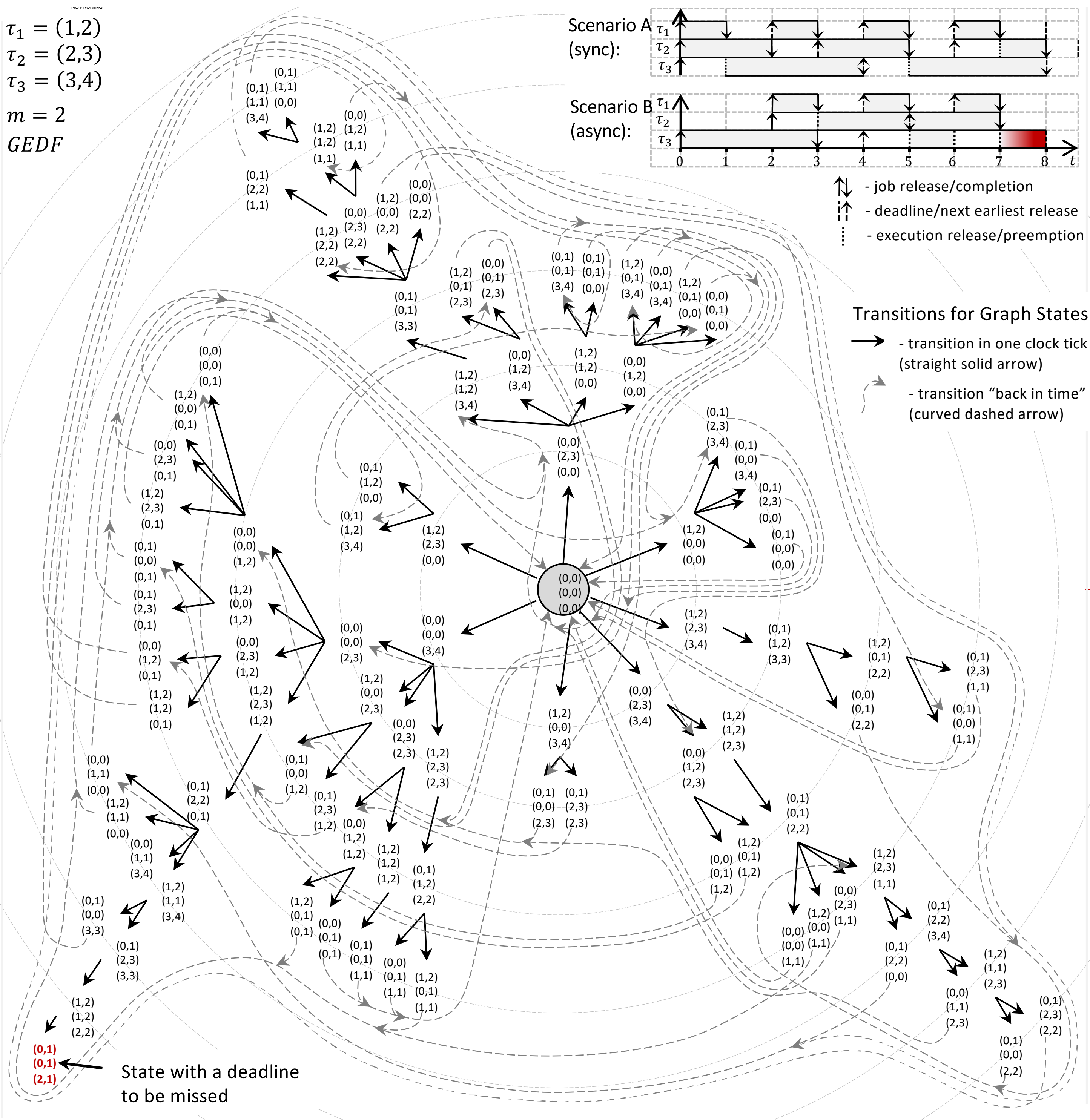


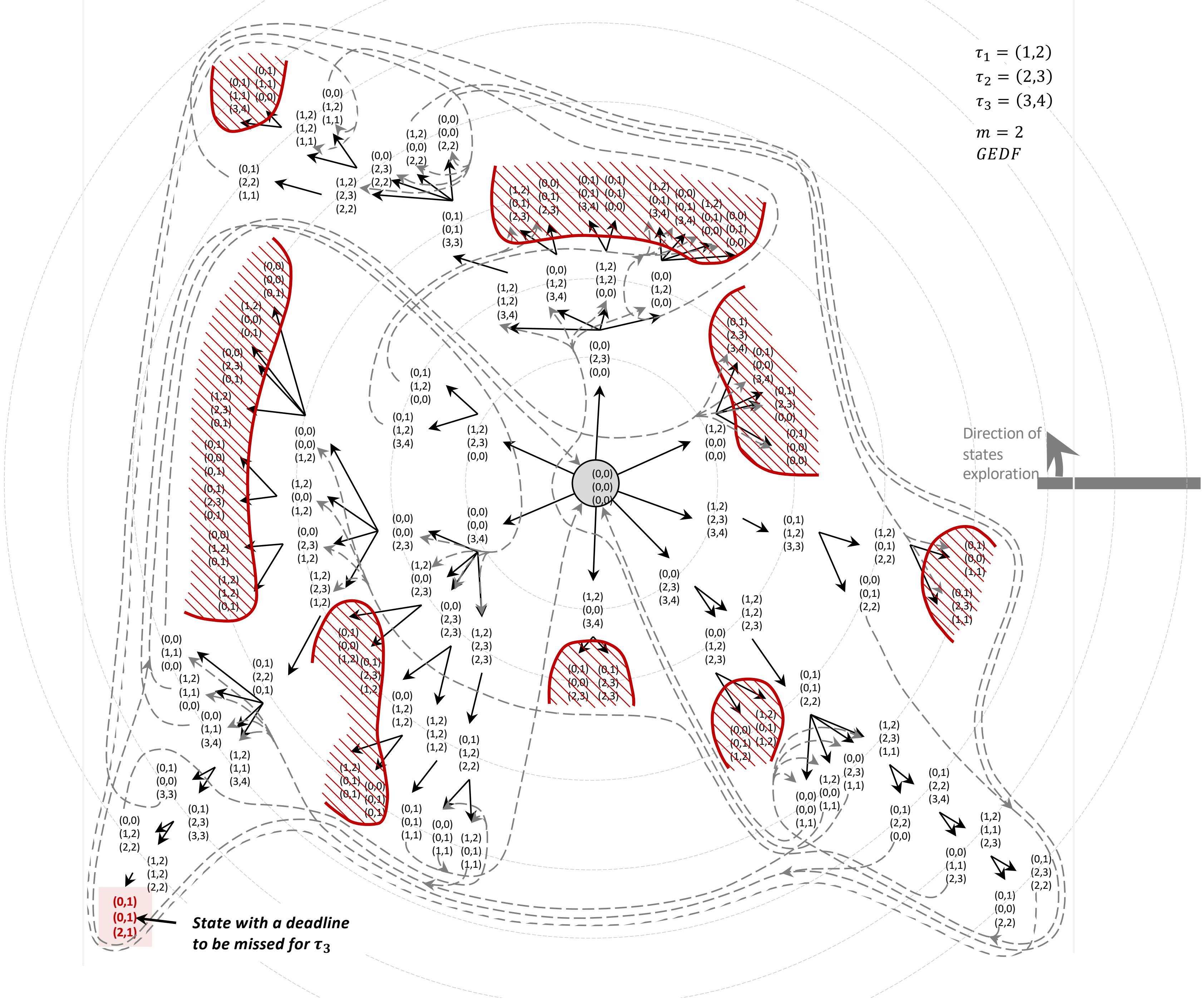



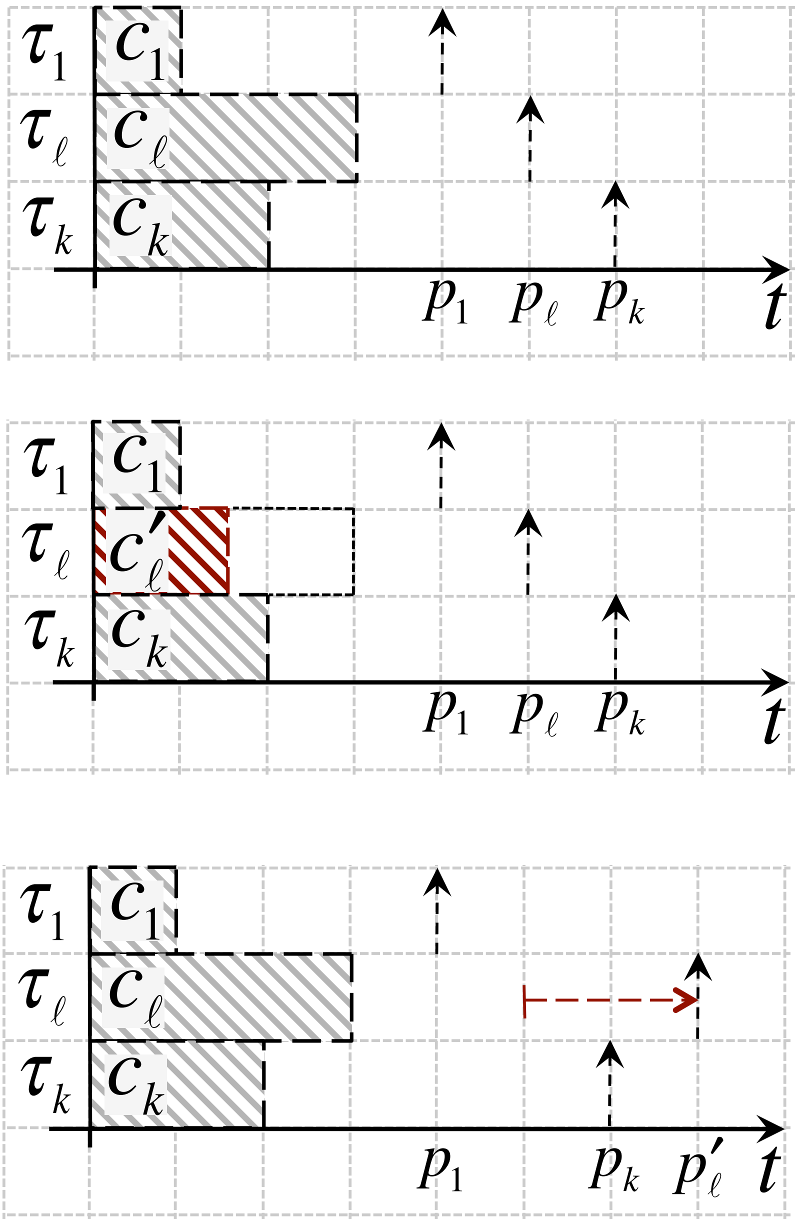
# Non-Pruned Graph

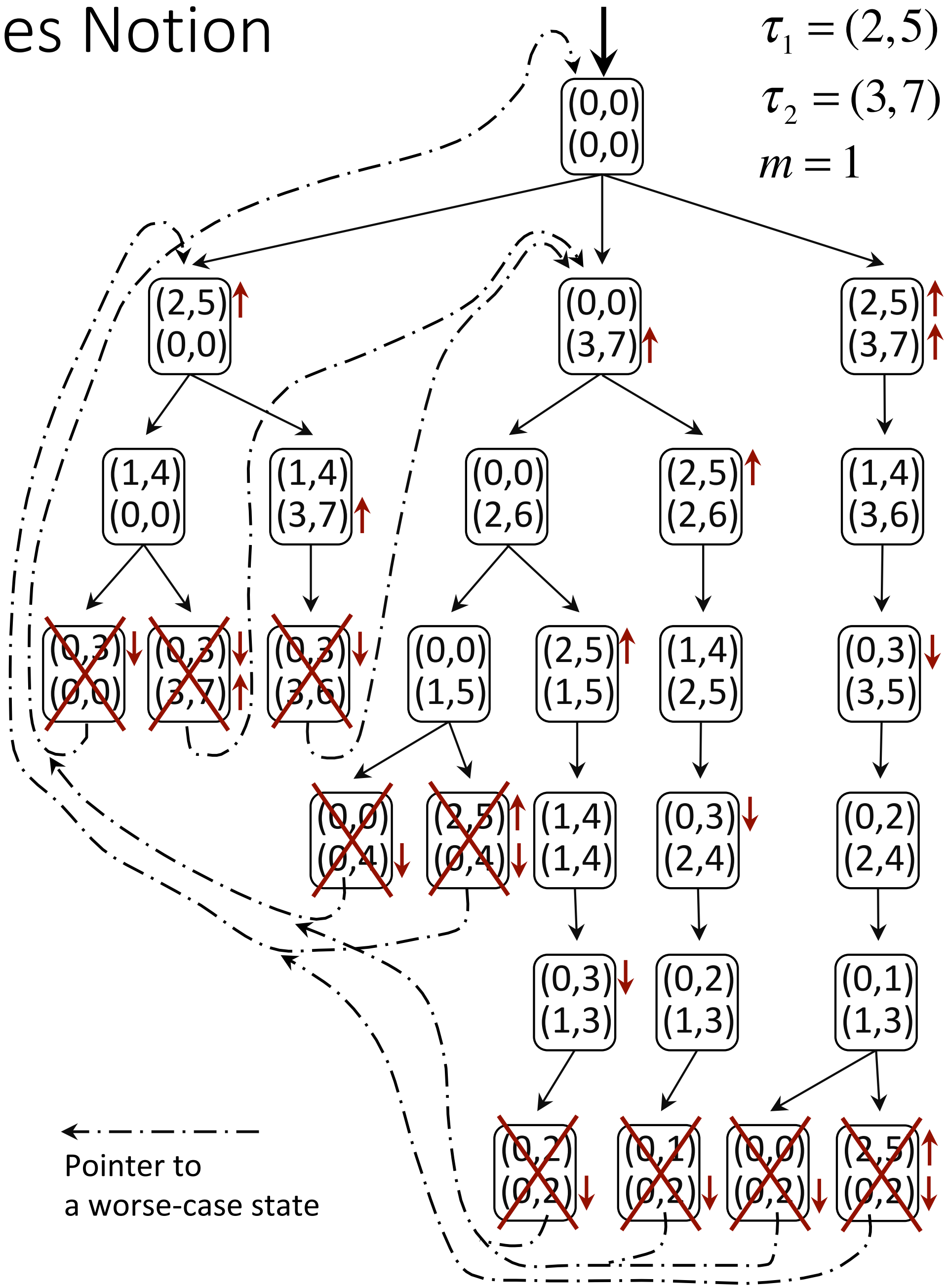


# Graph Pruned by the Worst-Case States Condition



# Pruning Based on the Worst-Case States Notion




**Problem: How to implement efficiently this pruning condition based on worst-case states**

**Map for**
**Storing Visiting Graph States**

Aims at facilitating
the implementation of
the worst-case states
pruning condition

| $B^c$ | $B^p$ | $B^{p,1}$ | $\sum_{i=1}^{k} c_i$ | $\sum_{i=1}^{k} p_i$ | $p_1$ | $p_2$ | $c_1$ | $c_2$ |
|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| 01 | 01 | 01 | 1 | 5 | 0 | 5 | 0 | 1 |
| | | | 2 | 6 | 0 | 6 | 0 | 2 |
| | | | 3 | 7 | 0 | 7 | 0 | 3 |
| | 11 | 00 | 1 | 4 | 1 | 3 | 0 | 1 |
| | | | | 5 | 2 | 3 | 0 | 1 |
| | | 01 | 2 | 6 | 2 | 4 | 0 | 2 |
| | | 10 | 1 | 6 | 3 | 3 | 0 | 1 |
| | | 11 | 2 | 7 | 3 | 4 | 0 | 2 |
| | | | 3 | 8 | 3 | 5 | 0 | 3 |
| 10 | 10 | 10 | 1 | 4 | 4 | 0 | 1 | 0 |
| | | | 2 | 5 | 5 | 0 | 2 | 0 |
| 11 | 11 | 11 | 2 | 8 | 4 | 4 | 1 | 1 |
| | | | 3 | 9 | 4 | 5 | 1 | 2 |
| | | | | 10 | 5 | 5 | 2 | 1 |
| | | | 4 | 10 | 4 | 6 | 1 | 3 |
| | | | | 11 | 4 | 7 | 1 | 3 |
| | | | | | 5 | 6 | 2 | 2 |
| | | | 5 | 12 | 5 | 7 | 2 | 3 |

# An Exact Schedulability Test for GFP (Pseudo-Code)

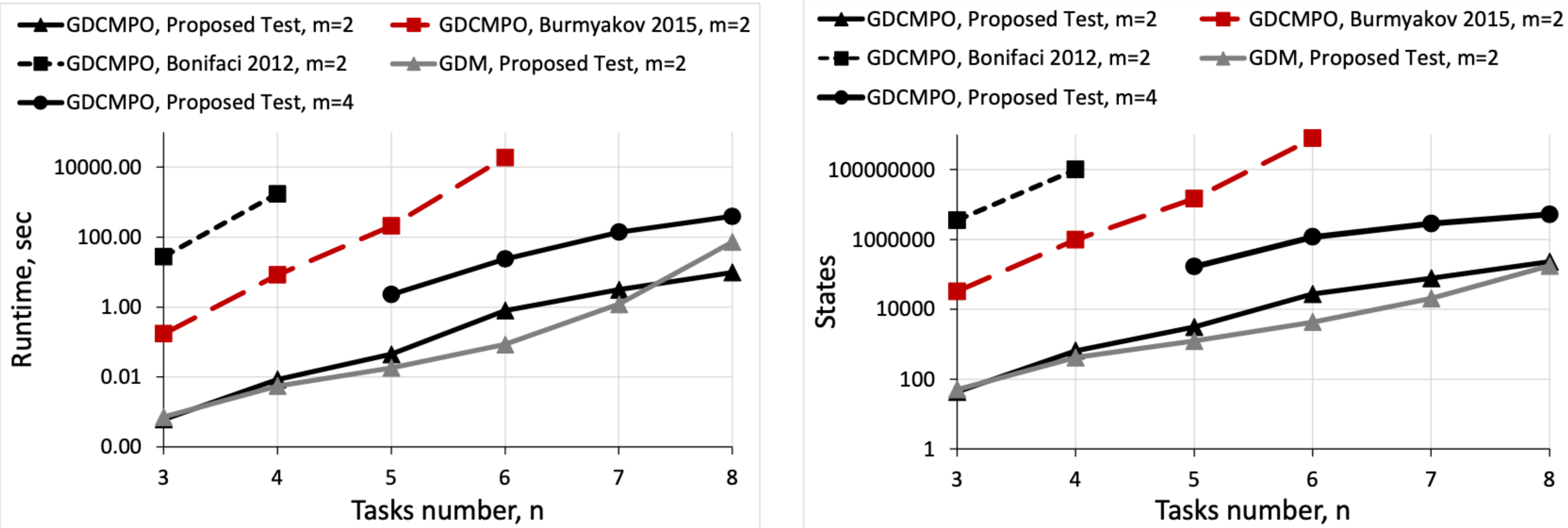
**Algorithm 1** Our pruning-based exact schedulability test

**function** EXACTSCHEDULABILITYTEST<br>
    $\mathcal{V} \leftarrow \varnothing$ ▷ initialize $\mathcal{V}$<br>
    $\mathcal{L} \leftarrow \{\mathbf{0}\}$ ▷ initialize $\mathcal{L}$<br>
    **while** $\mathcal{L} \neq \varnothing$ **do** ▷ loop until states need to be visited<br>
        $x \leftarrow \text{FETCH}(\mathcal{L})$<br>
        $\mathcal{L}' \leftarrow \{x' \in \mathcal{X} : x' = \Phi(x, r), r \in \mathcal{R}(x)\}$ ▷ $\mathcal{R}(x)$ from (8), $\Phi$ from (9)<br>
        $\mathcal{L}' \leftarrow \text{PRUNENEXTSTATES}(\mathcal{L}', \mathcal{V})$ ▷ see Section 4<br>
        **for** each $x' \in \mathcal{L}'$ **do**<br>
            **if** $x' \notin \mathcal{F}$ **then** ▷ deadline miss<br>
                **return** Unschedulable<br>
            **end if**<br>
            $\mathcal{V} \leftarrow \mathcal{V} \cup \{x'\}$<br>
            $\mathcal{L} \leftarrow \text{INSERT}(\mathcal{L}, x')$<br>
        **end for**<br>
    **end while**<br>
    **return** Schedulable<br>
**end function**

Some Evaluation Results for
Runtime and Memory Consumption

Table 1: Key parameters: default values

| Settings | | |
|---|---|---|
| Number of cores, $m$ | 2 | 4 |
| Number of heavy tasks, $n_{\text{heavy}}$ | arbitrary | 5[3] |
| Task set utilization, $U_{\mathcal{T}}$ | 1.5 | 3.3 |
| Number of tasks, $n$ | 5 | 8 |
| Range for task periods, $P_i$ | [5, 25] | [5, 20] |
| Ratio of task periods, $P_{\max}/P_{\min}$ | 5 | 4 |
| Ratio of task periods to deadlines | 1 | 1 |



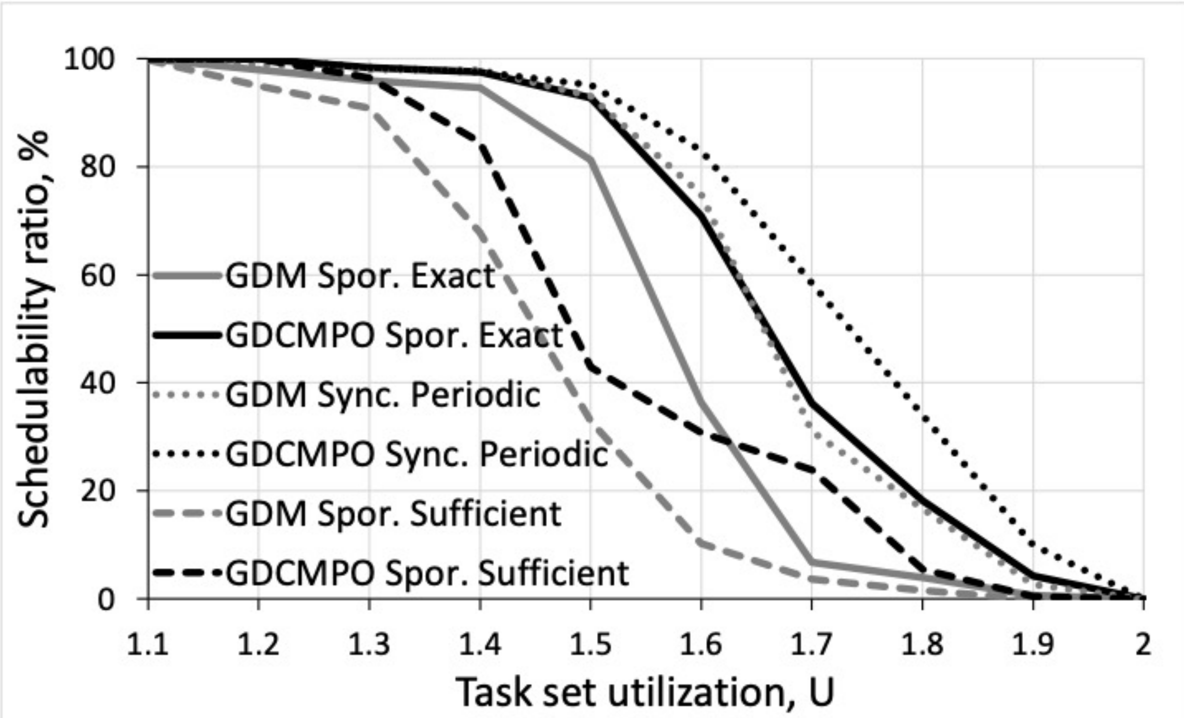

(a) Runtime

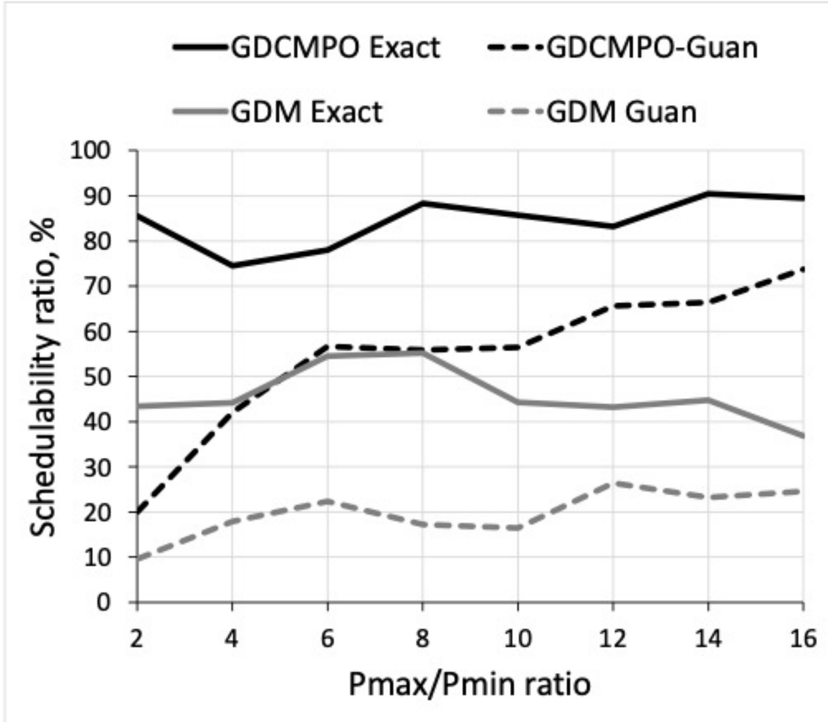
(b) States

# Some Evaluation Results for Achievable Schedulability Ratios

Table 1: Key parameters: default values

| Settings | | |
|---|---|---|
| Number of cores, $m$ | 2 | 4 |
| Number of heavy tasks, $n_{\mathsf{heavy}}$ | arbitrary | 5[3] |
| Task set utilization, $U_{\mathcal{T}}$ | 1.5 | 3.3 |
| Number of tasks, $n$ | 5 | 8 |
| Range for task periods, $P_i$ | [5, 25] | [5, 20] |
| Ratio of task periods, $P_{\mathsf{max}}/P_{\mathsf{min}}$ | 5 | 4 |
| Ratio of task periods to deadlines | 1 | 1 |

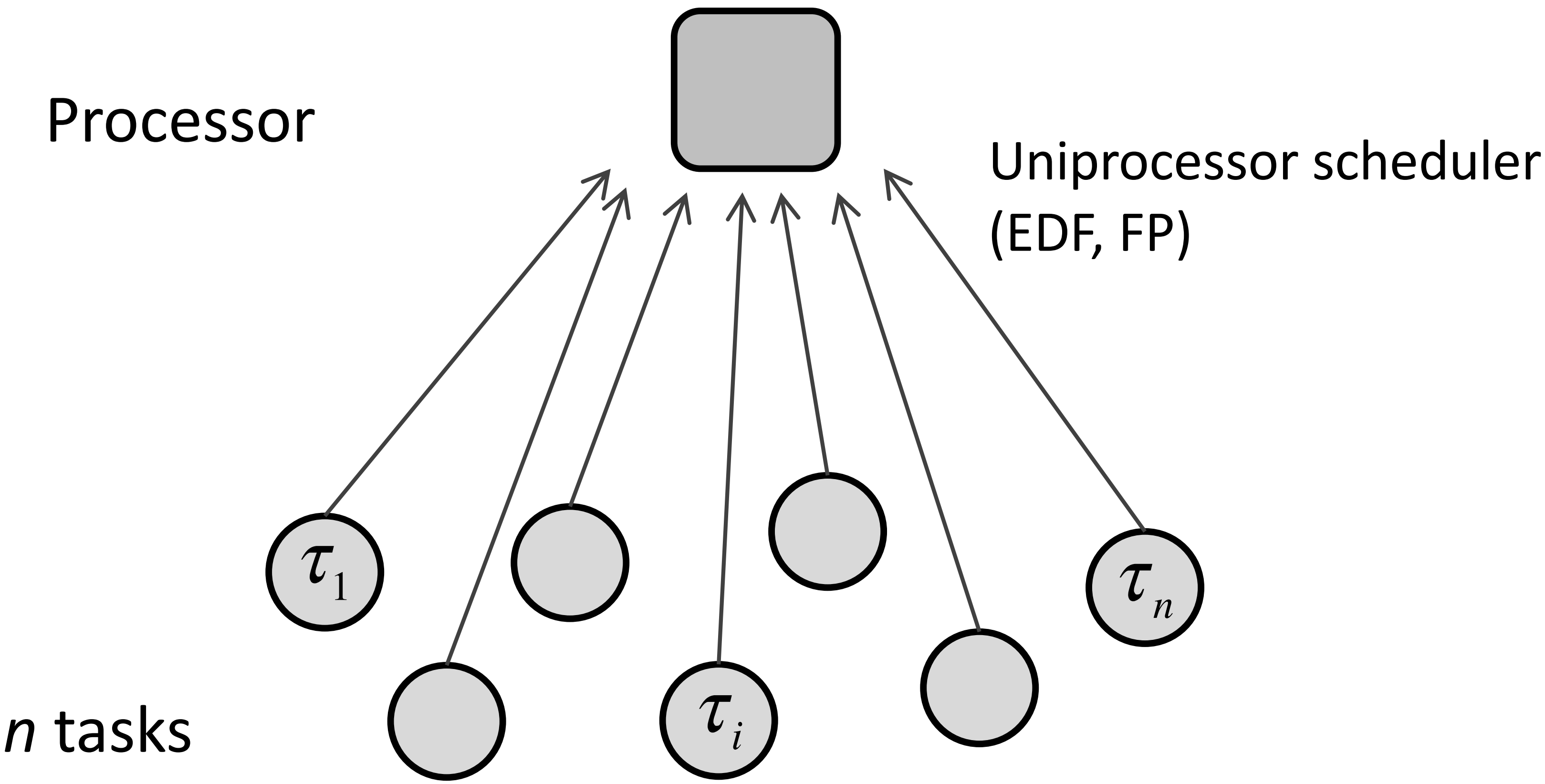


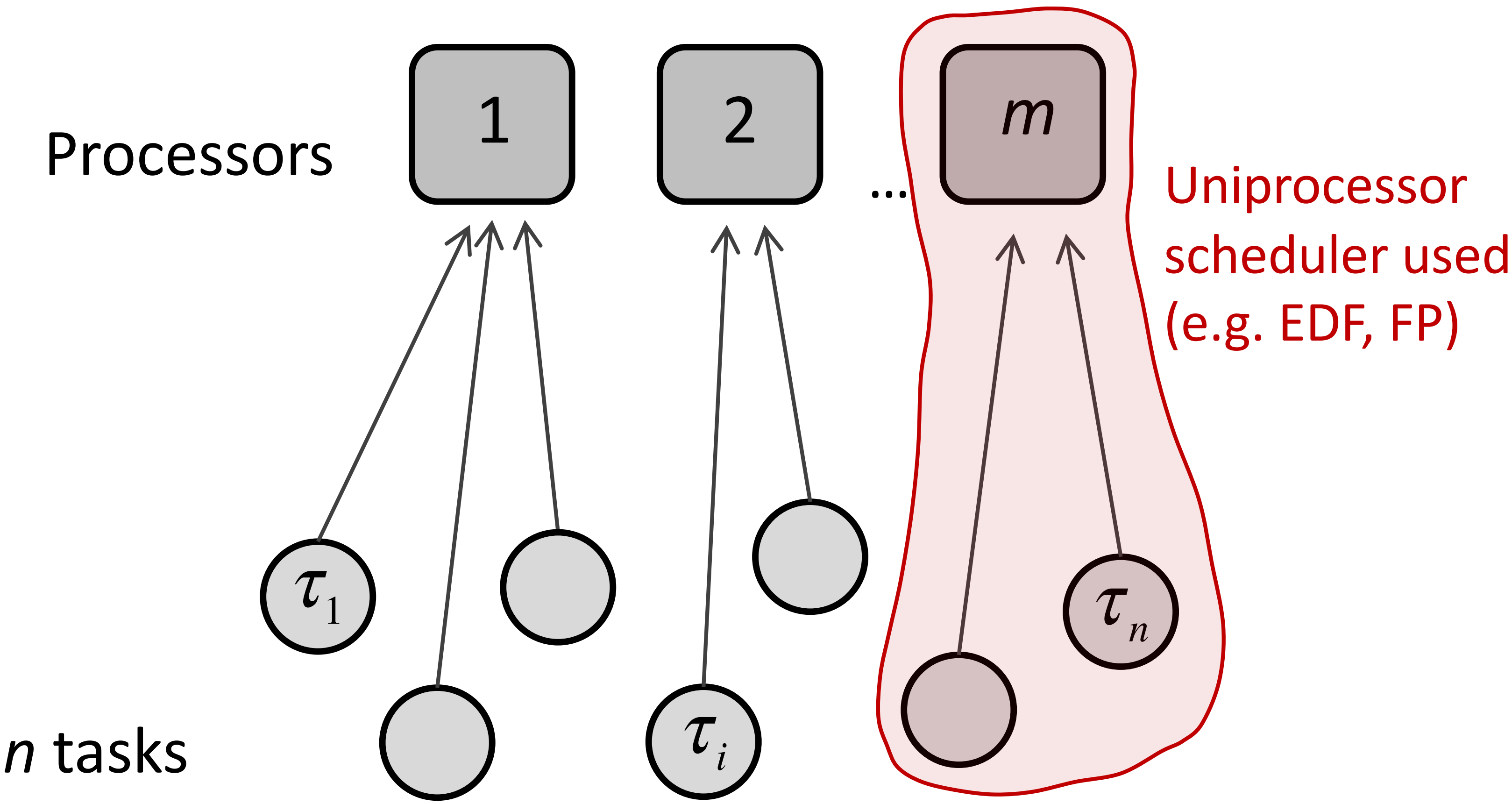
# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**:
  global, partitioned, and semi-partitioned scheduling;

# Uniprocessor Scheduling



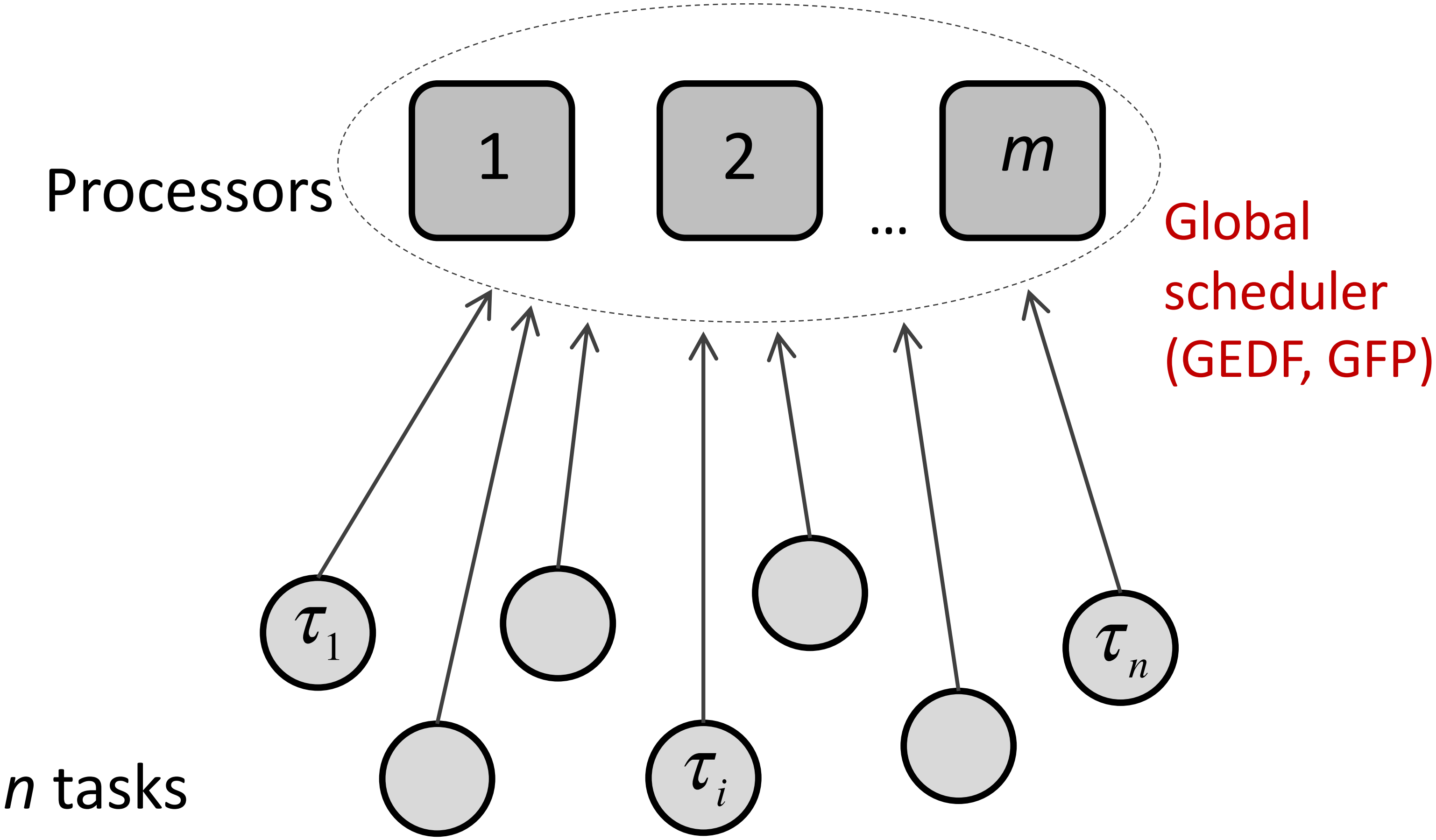
# Multiprocessor Scheduling




"Partitioned" multiprocessor scheduling:
Each task is statically assigned to some processor

Open Problem:
How to partition tasks between processors?

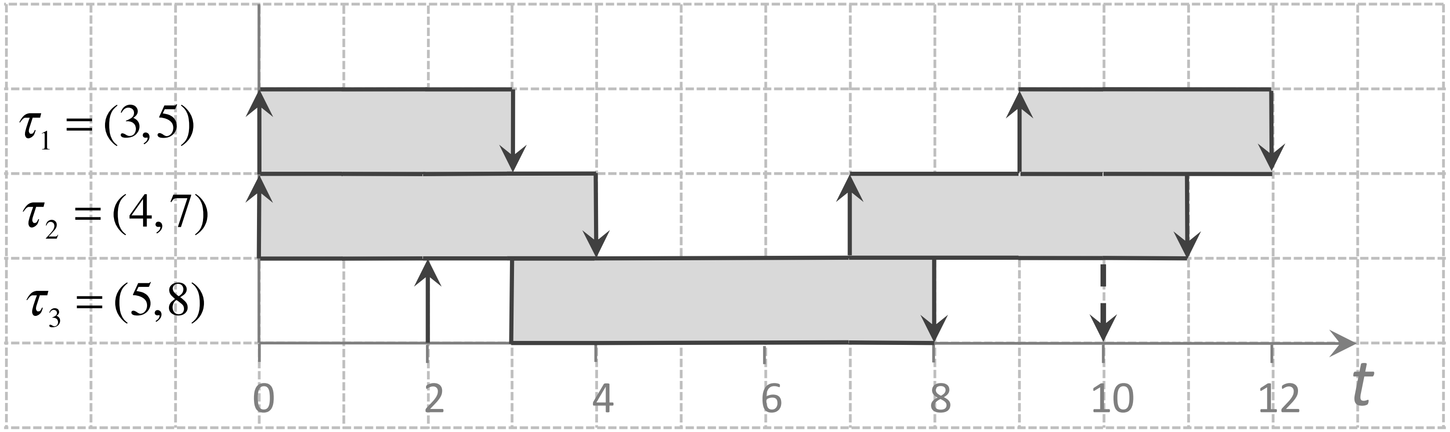
# Multiprocessor Scheduling




Global scheduling:
Tasks migrate between all processors

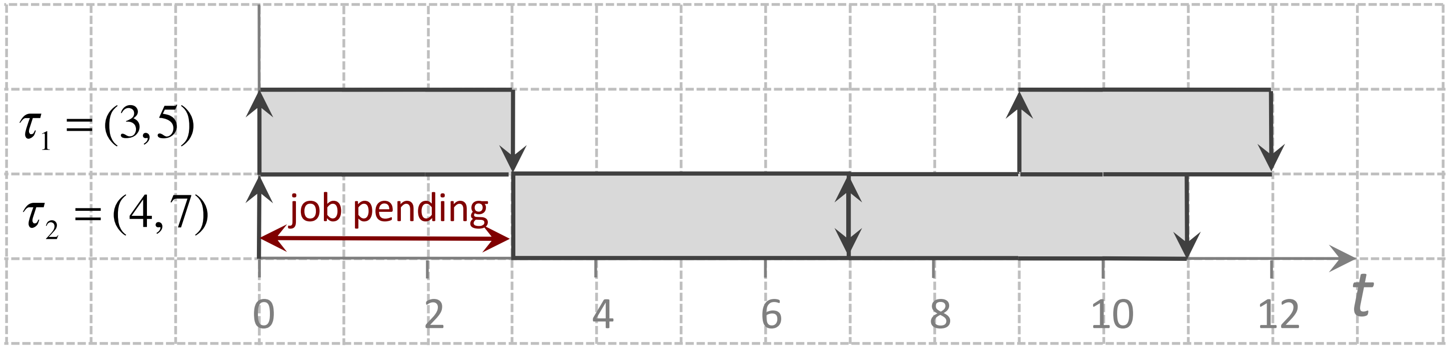
# Global FP (GFP) schedule

m=2
processors

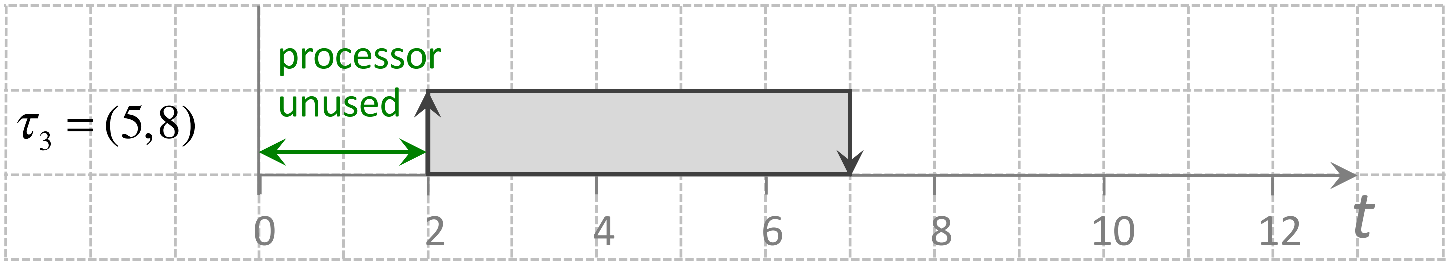



# Partitioned FP (PFP) schedule

Proc. 1
$\{\tau_1, \tau_2\}$

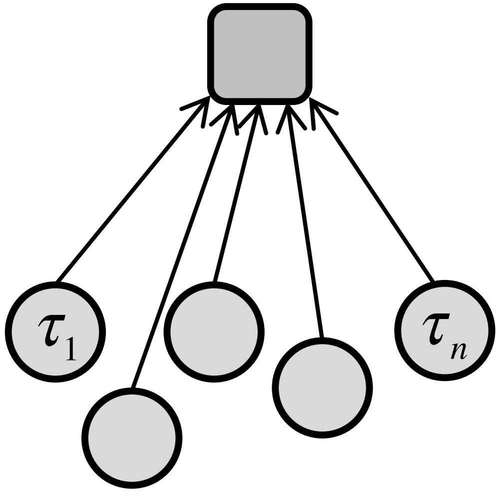



Proc. 2
$\{\tau_3\}$



# Scheduling

## Uniprocessor



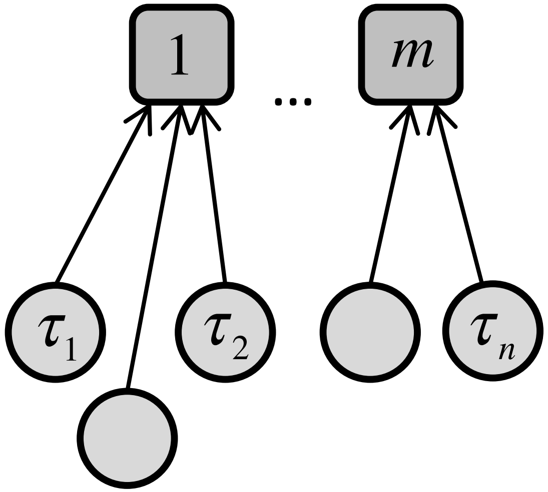

Well studied:
- efficient sched. tests;
- optimal schedulers.

[Liu and Layland 1973]

[Joseph and Pandya 1986]

## Multiprocessor

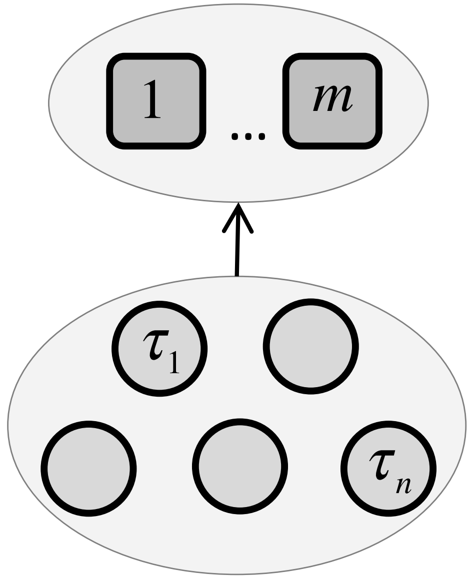
### *Partitioned*




Problem converges to uniprocessor scheduling

No optimal strategy known to partition tasks

### *Global*




Poorly understood:
- no efficient optimal schedulers;
- no exact schedulability test;
- no optimal priority assignment for GFP

[Davis and Burns 2010]

# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**:
  global, partitioned, and semi-partitioned scheduling;

# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**: global, partitioned, and semi-partitioned scheduling;

**Table 2: Key parameters: default values**

| Settings | | |
|---|---|---|
| Number of processors, $m$ | 2 | 4 |
| Number of tasks, $n$ | 7 | 14 |
| Number of heavy tasks, $n_{\text{heavy}}$ | 2 | 4 |
| Task set utilization, $U_{\mathcal{T}}$ (excl. exps. for varying $P_i/D_i$ ratio) | 1.75 | 3 |
| Task set density, $D_{\mathcal{T}}$ (for exps. with varying $P_i/D_i$ ratio) | 2 | 4 |
| Ratio between the maximum and minimum task deadlines, $D_{\text{max}}/D_{\text{min}}$ | 20 | 5 |
| Range for task deadlines, $D_i$ | [5; 120] | [7; 35] |
| Ratio between task periods and deadlines, $P_i/D_i$ | 1 | 1 |

# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**: global, partitioned, and semi-partitioned scheduling;

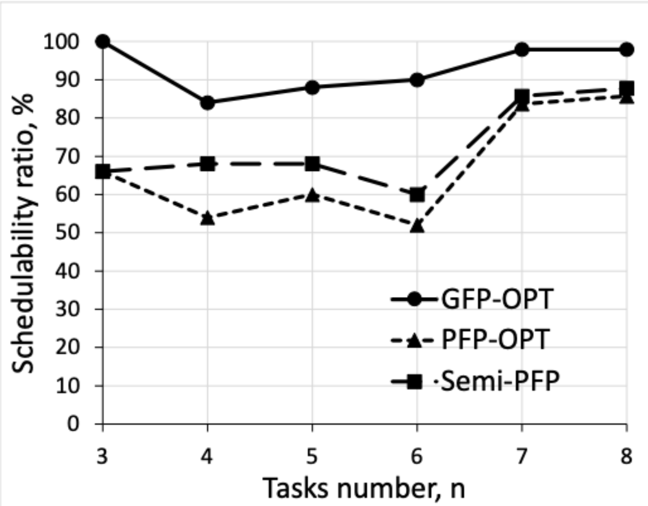
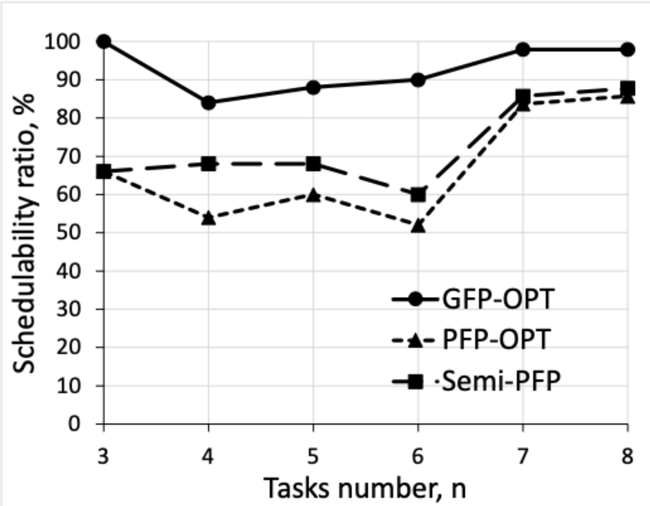



**Table 2: Key parameters: default values**

| Settings | | |
|---|---|---|
| Number of processors, $m$ | 2 | 4 |
| Number of tasks, $n$ | 7 | 14 |
| Number of heavy tasks, $n_{\text{heavy}}$ | 2 | 4 |
| Task set utilization, $U_{\mathcal{T}}$ (excl. exps. for varying $P_i/D_i$ ratio) | 1.75 | 3 |
| Task set density, $D_{\mathcal{T}}$ (for exps. with varying $P_i/D_i$ ratio) | 2 | 4 |
| Ratio between the maximum and minimum task deadlines, $D_{\max}/D_{\min}$ | 20 | 5 |
| Range for task deadlines, $D_i$ | [5; 120] | [7; 35] |
| Ratio between task periods and deadlines, $P_i/D_i$ | 1 | 1 |

# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**: global, partitioned, and semi-partitioned scheduling;



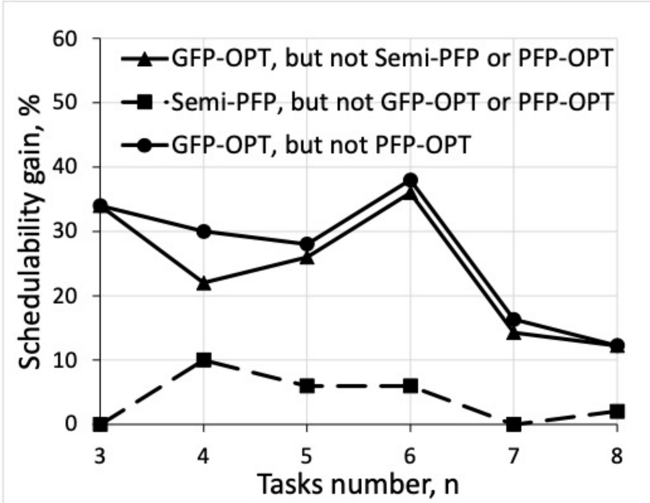
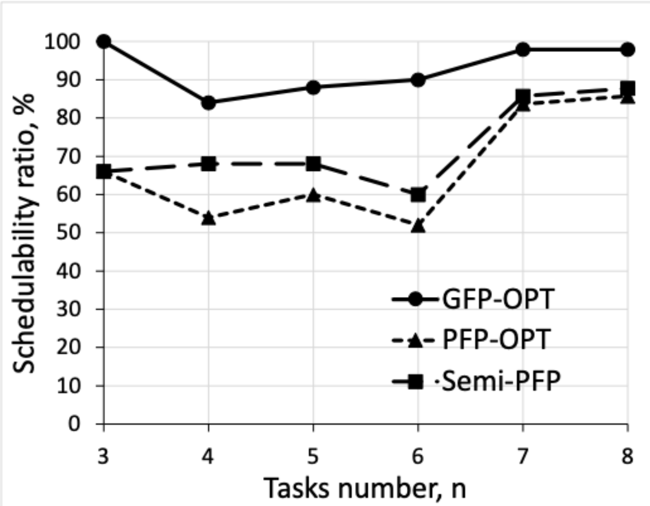



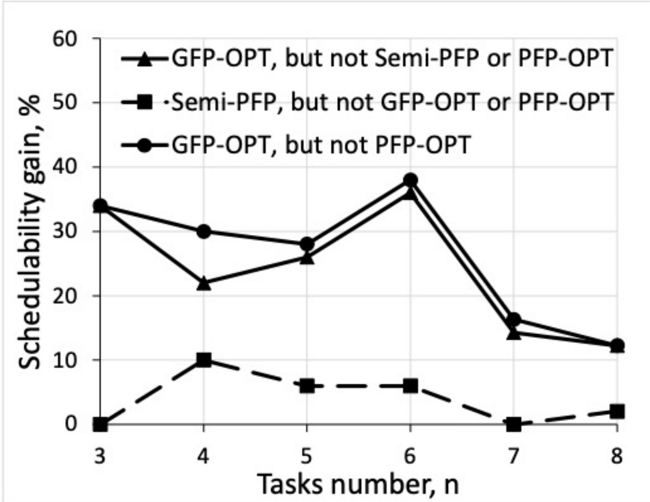

**Table 2: Key parameters: default values**

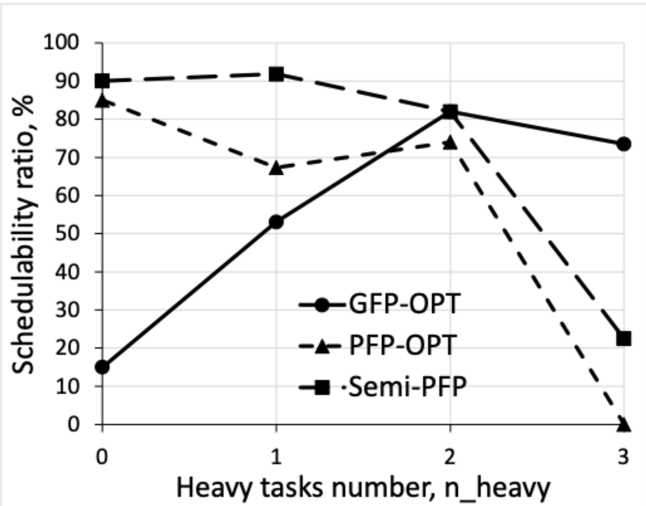
| Settings | | |
|---|---|---|
| Number of processors, $m$ | 2 | 4 |
| Number of tasks, $n$ | 7 | 14 |
| Number of heavy tasks, $n_{\text{heavy}}$ | 2 | 4 |
| Task set utilization, $U_{\mathcal{T}}$ (excl. exps. for varying $P_i/D_i$ ratio) | 1.75 | 3 |
| Task set density, $D_{\mathcal{T}}$ (for exps. with varying $P_i/D_i$ ratio) | 2 | 4 |
| Ratio between the maximum and minimum task deadlines, $D_{\max}/D_{\min}$ | 20 | 5 |
| Range for task deadlines, $D_i$ | [5; 120] | [7; 35] |
| Ratio between task periods and deadlines, $P_i/D_i$ | 1 | 1 |

# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**: global, partitioned, and semi-partitioned scheduling;



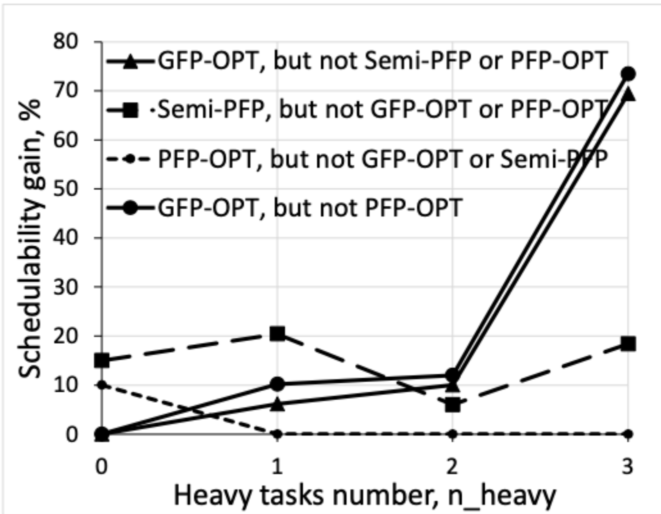




**Table 2: Key parameters: default values**

| Settings | | |
|---|---|---|
| Number of processors, $m$ | 2 | 4 |
| Number of tasks, $n$ | 7 | 14 |
| Number of heavy tasks, $n_{\text{heavy}}$ | 2 | 4 |
| Task set utilization, $U_{\mathcal{T}}$ (excl. exps. for varying $P_i/D_i$ ratio) | 1.75 | 3 |
| Task set density, $D_{\mathcal{T}}$ (for exps. with varying $P_i/D_i$ ratio) | 2 | 4 |
| Ratio between the maximum and minimum task deadlines, $D_{\max}/D_{\min}$ | 20 | 5 |
| Range for task deadlines, $D_i$ | [5; 120] | [7; 35] |
| Ratio between task periods and deadlines, $P_i/D_i$ | 1 | 1 |

[Burmyakov & Nikolic 2021]

# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**:
  global, partitioned, and semi-partitioned scheduling;

- **Efficiency analysis of different priority assignments**:
  RM, DM, DCMPO, DkC, DC;

(1) DCMPO assigns a higher priority to task $\tau_i$ with a lower value of $(D_i - C_i)$;

(2) DkC assigns a higher to task $\tau_i$ with a lower value of $(D_i - k \times C_i)$, where $k$ is computed by

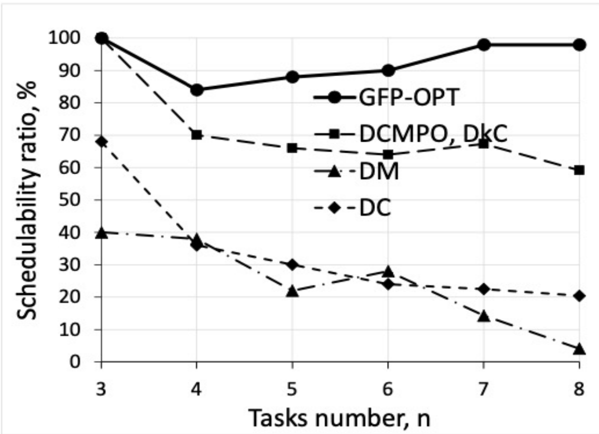
$$k = \frac{m - 1 + \sqrt{5m^2 - 6m + 1}}{2m}$$

(3) DC assigns a higher priority to a task with a higher value of $C_i/D_i$

# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**: global, partitioned, and semi-partitioned scheduling;

- **Efficiency analysis of different priority assignments**: RM, DM, DCMPO, DkC, DC;



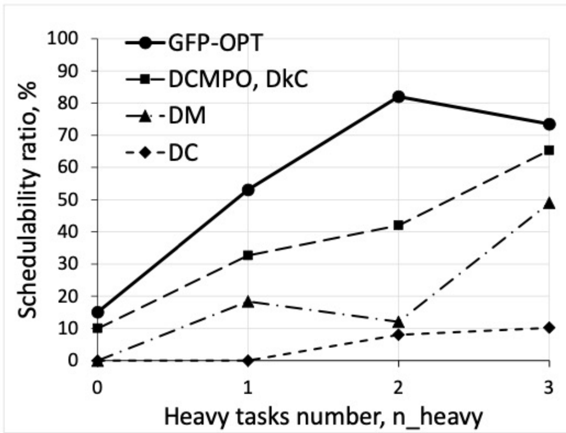



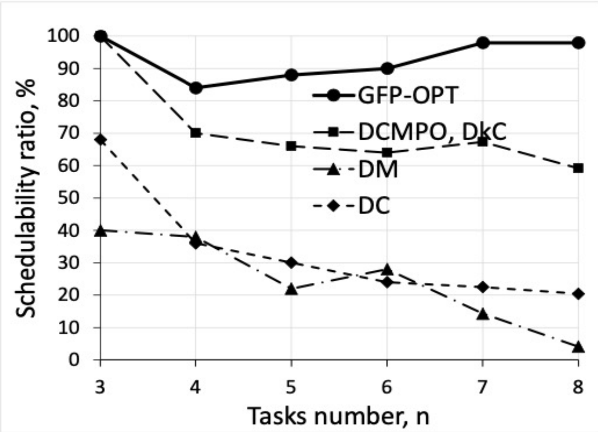

**Table 2: Key parameters: default values**

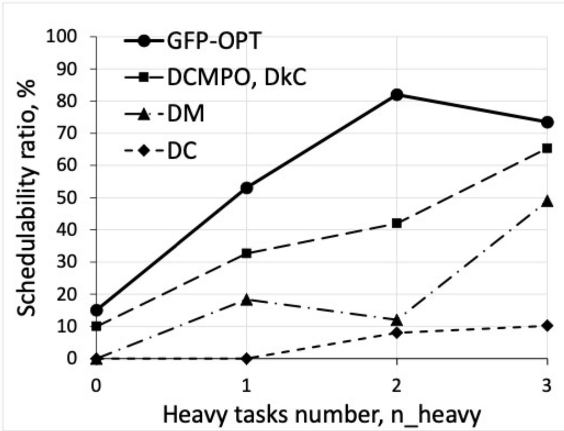
| Settings | | |
|---|---|---|
| Number of processors, $m$ | 2 | 4 |
| Number of tasks, $n$ | 7 | 14 |
| Number of heavy tasks, $n_{\text{heavy}}$ | 2 | 4 |
| Task set utilization, $U_{\mathcal{T}}$ (excl. exps. for varying $P_i/D_i$ ratio) | 1.75 | 3 |
| Task set density, $D_{\mathcal{T}}$ (for exps. with varying $P_i/D_i$ ratio) | 2 | 4 |
| Ratio between the maximum and minimum task deadlines, $D_{\max}/D_{\min}$ | 20 | 5 |
| Range for task deadlines, $D_i$ | [5; 120] | [7; 35] |
| Ratio between task periods and deadlines, $P_i/D_i$ | 1 | 1 |

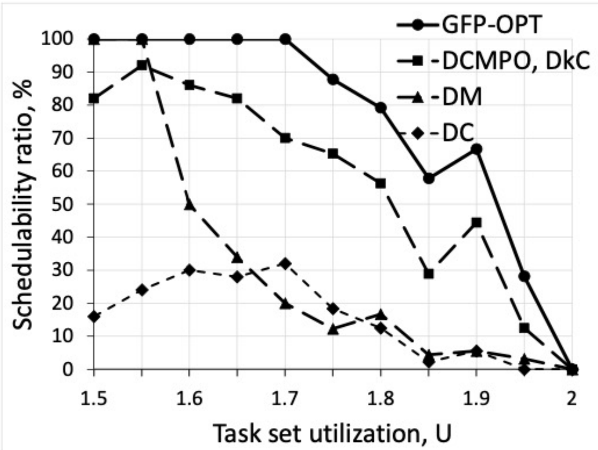
# Application of Exact Schedulability Test

- **Performance evaluation of different scheduling strategies**: global, partitioned, and semi-partitioned scheduling;

- **Efficiency analysis of different priority assignments**: RM, DM, DCMPO, DkC, DC;



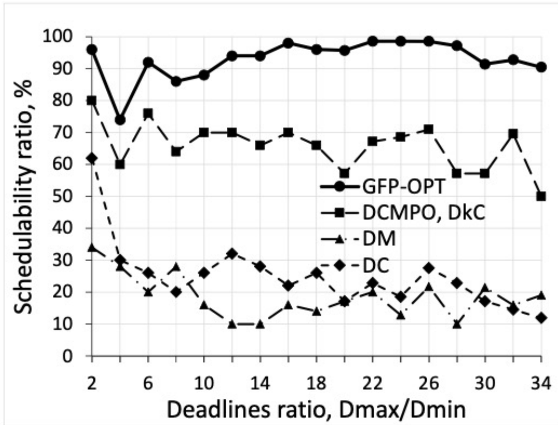




**Table 2: Key parameters: default values**

| Settings | | |
|---|---|---|
| Number of processors, $m$ | 2 | 4 |
| Number of tasks, $n$ | 7 | 14 |
| Number of heavy tasks, $n_{\text{heavy}}$ | 2 | 4 |
| Task set utilization, $U_{\mathcal{T}}$ (excl. exps. for varying $P_i/D_i$ ratio) | 1.75 | 3 |
| Task set density, $D_{\mathcal{T}}$ (for exps. with varying $P_i/D_i$ ratio) | 2 | 4 |
| Ratio between the maximum and minimum task deadlines, $D_{\max}/D_{\min}$ | 20 | 5 |
| Range for task deadlines, $D_i$ | [5; 120] | [7; 35] |
| Ratio between task periods and deadlines, $P_i/D_i$ | 1 | 1 |

[Burmyakov & Nikolic 2021]