Schedulability Analysis of the **Linux Push and Pull Scheduler** with **Arbitrary Processor Affinities**

Arpan Gujarati, Felipe Cerqueira, and Björn Brandenburg

Max Planck Institute for Software Systems

ECRTS 2013
9-12 July
Paris, France

Multiprocessor real-time scheduling **theory**

Global — **Unrestricted** migration

Partitioned — **No** migration

Multiprocessor real-time scheduling **theory**

Global — **Unrestricted** migration

Partitioned — **No** migration

Clustered — Tasks can migrate only to processors **within** its **cluster**

Semi-partitioned — Only **some tasks** allowed to migrate

Meanwhile in **practice...**

CPU affinity interface in Linux
(specify the CPUs on which a task can execute)

```
int sched_setaffinity(pid_t pid, size_t cpusetsize,
                      cpu_set_t *mask);

int sched_getaffinity(pid_t pid, size_t cpusetsize,
                      cpu_set_t *mask);
```

Meanwhile in **practice...**

CPU affinity interface in Linux
(specify the CPUs on which a task can execute)

```
int sched_setaffinity(pid_t pid, size_t cpusetsize,
                      cpu_set_t *mask);

int sched_getaffinity(pid_t pid, size_t cpusetsize,
                      cpu_set_t *mask);
```

- **Fine-grained** control over task migrations
- **A**rbitrary **P**rocessor **A**ffinity (APA)
- E.g., fault tolerance, security concerns

Meanwhile in **practice**...

CPU affinity interface in Linux
(specify the CPUs on which a task can execute)

```
int
in
```

Understand the CPU affinity interface from a **schedulability** point of view

- **A**rbitrary **P**rocessor **A**ffinity (APA)
- E.g., fault tolerance, security concerns

Objective

Is the APA interface just an implementation detail or does it have interesting **theoretical implications**?

Objective

Is the APA interface just an implementation detail or does it have interesting **theoretical implications**?

How can we derive schedulability **guarantees** for APA schedulers?

Objective

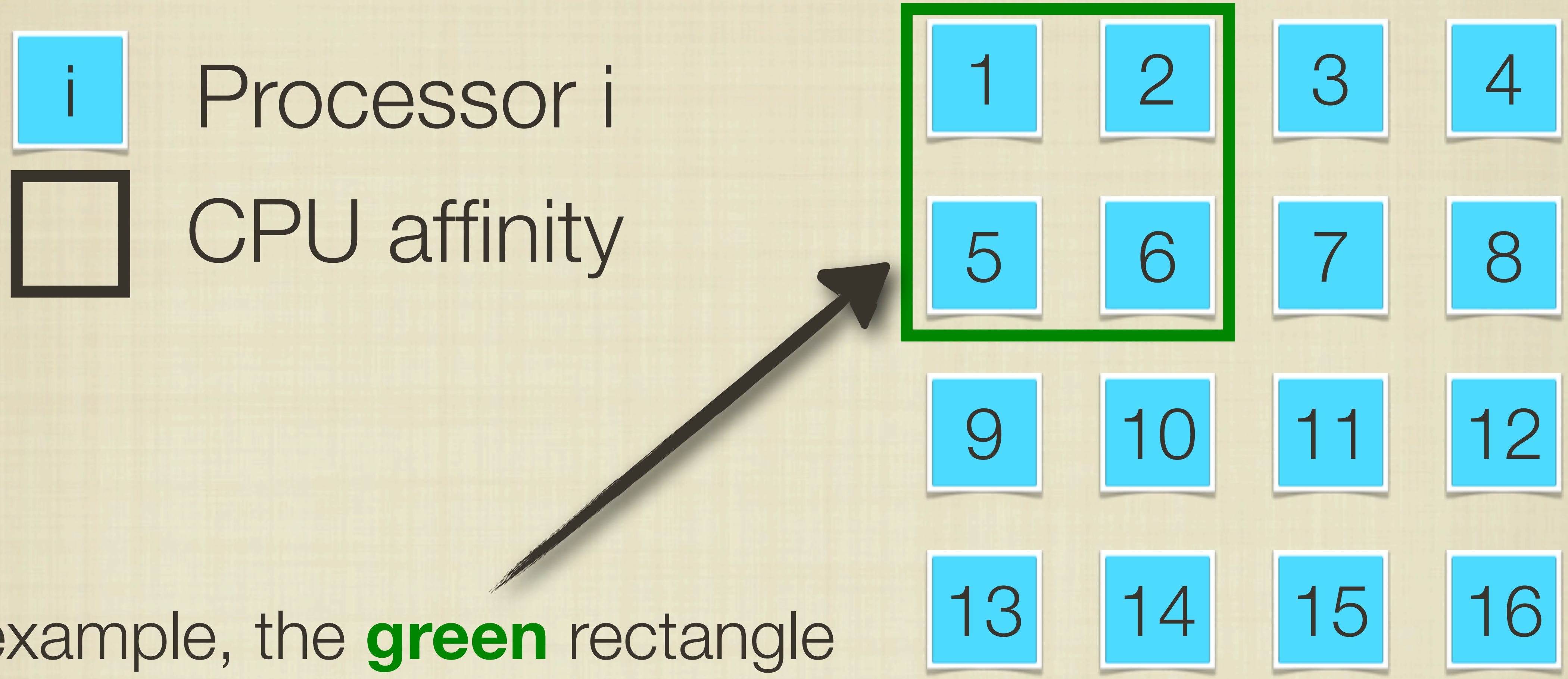
Is the APA interface just an implementation detail or does it have interesting **theoretical implications**?

How can we derive schedulability **guarantees** for APA schedulers?

Does APA scheduling help **improve** schedulability?

Illustrating CPU affinity interface

Illustrating CPU affinity interface

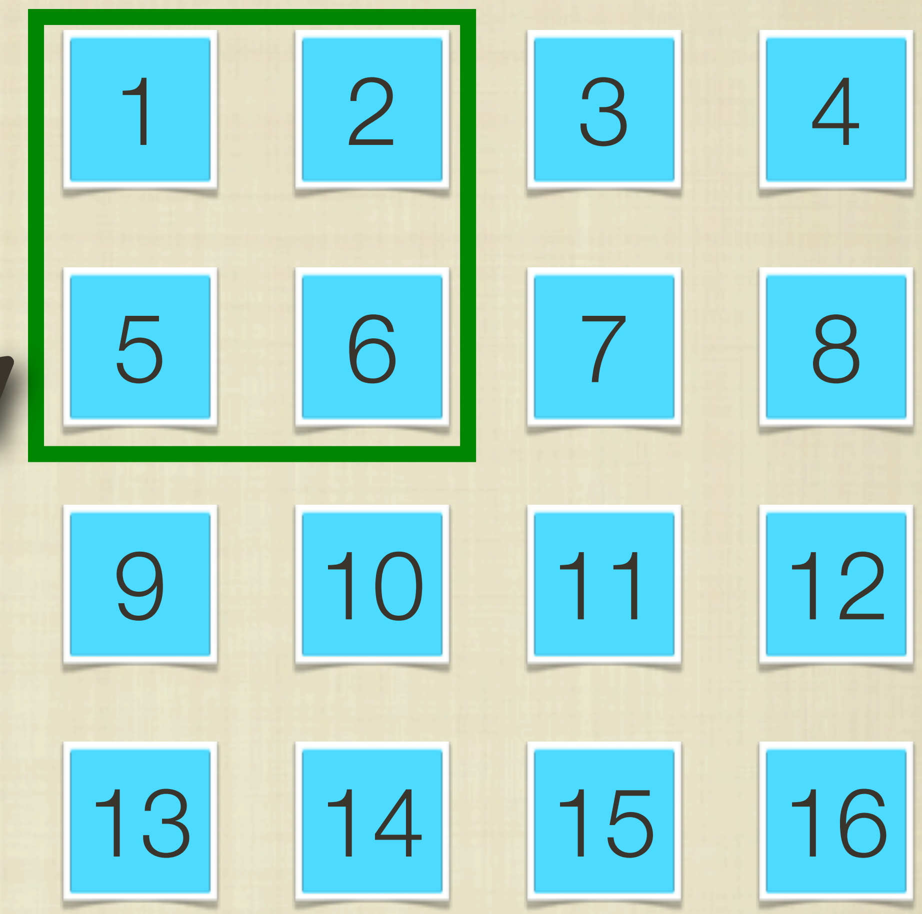


For example, the **green** rectangle indicates a CPU affinity **{1, 2, 5, 6}**

Illustrating CPU affinity interface

i Processor i

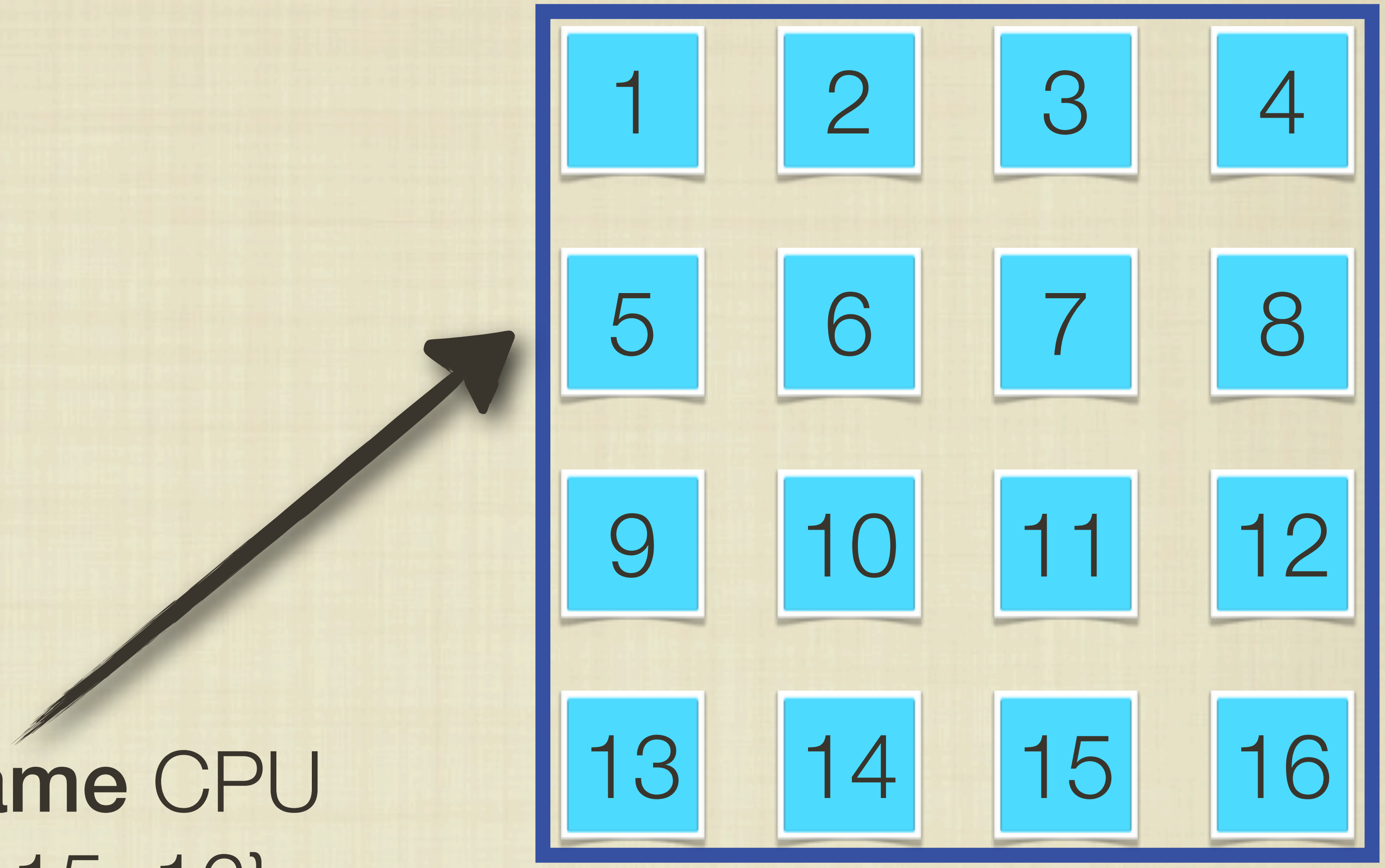
☐ CPU affinity



For example, the **green** rectangle indicates a CPU affinity **{1, 2, 5, 6}**

Can emulate **global, partitioned, clustered** scheduling

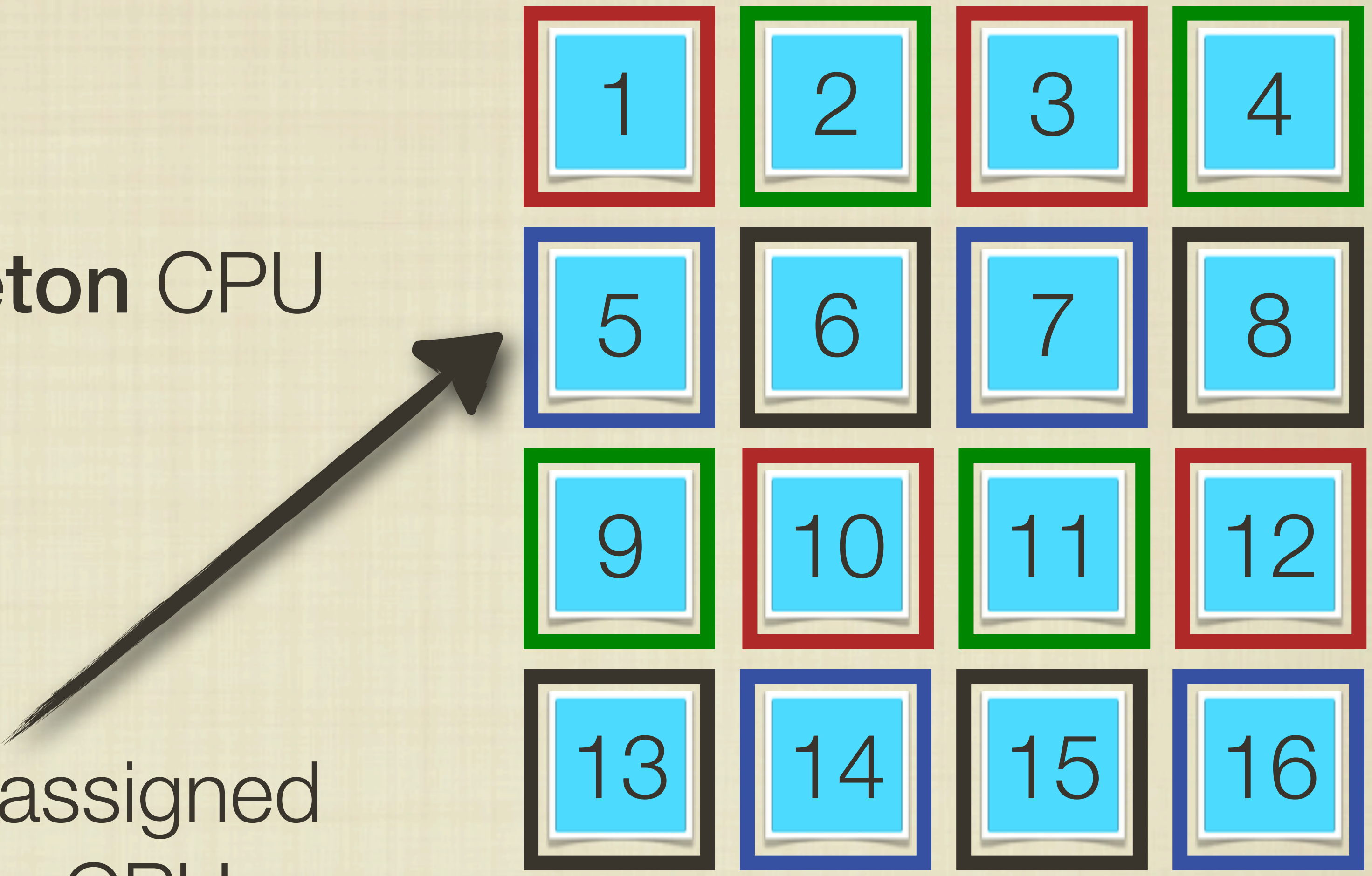
Emulating **global** scheduling



All tasks have the **same** CPU affinity: {1, 2, 3, ..., 15, 16}

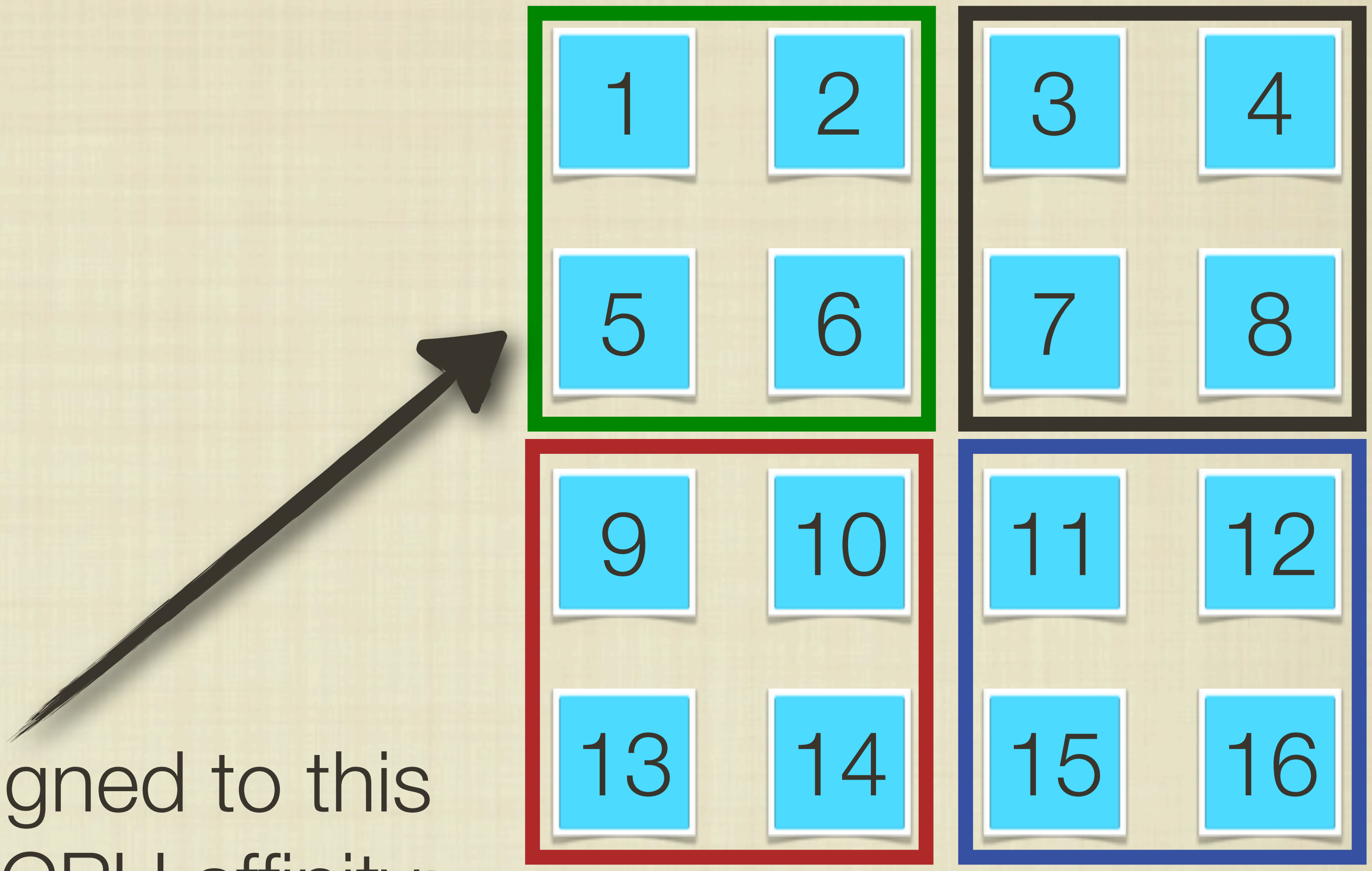
Emulating **partitioned** scheduling

All tasks have **singleton** CPU affinities.



For example, a task assigned to this partition has CPU affinity: {5}

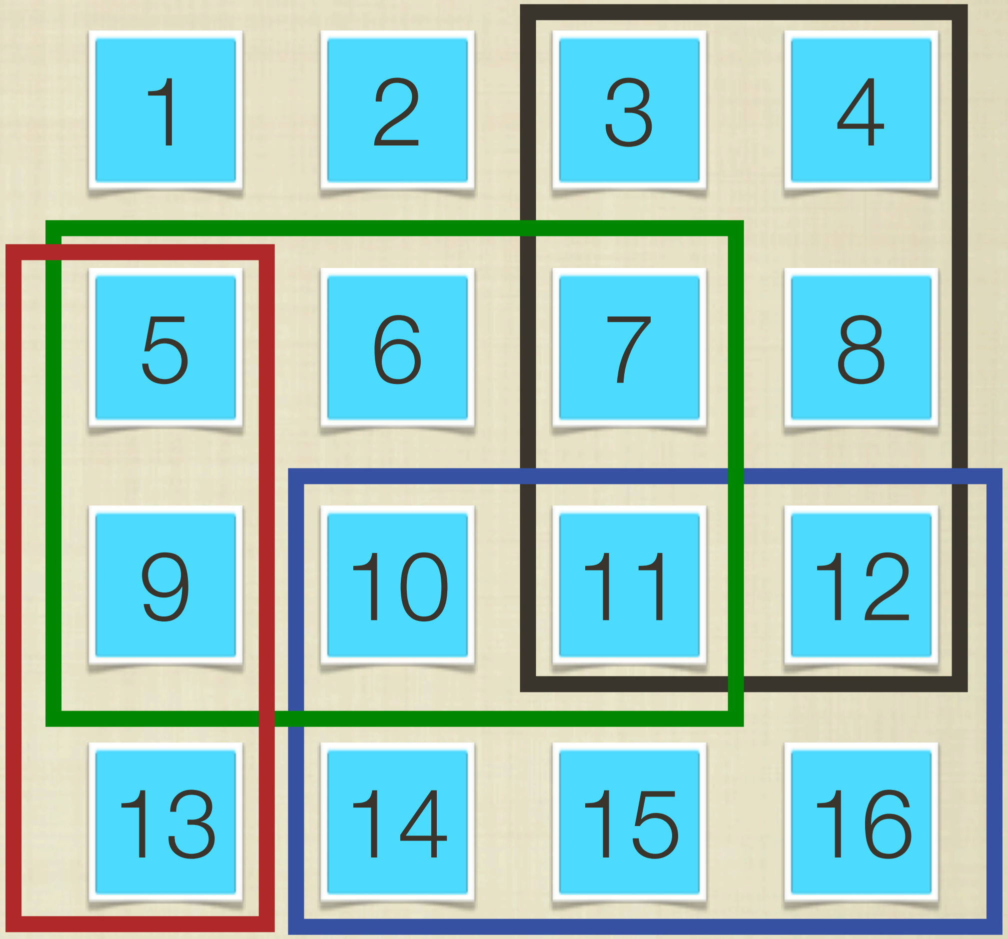
Emulating **clustered** scheduling



Example: all tasks assigned to this cluster have the same CPU affinity: {1, 2, 5, 6}

Distinct CPU affinity for each task

No existing schedulability analysis!



APA scheduler

Reference scheduler

Linux scheduler

Source-initiated **push** migrations

Target-initiated **pull** migrations

APA scheduler

A task is **not** scheduled only if **all processors in its affinity are busy** executing higher-priority tasks

Reference scheduler

Linux scheduler

Source-initiated **push** migrations

Target-initiated **pull** migrations

APA scheduler

A task is **not** scheduled only if **all processors in its affinity are busy** executing higher-priority tasks

The Linux scheduler **never violates** a task's affinity

Target-initiated **pull** migrations

APA scheduler

A task is **not** scheduled only if **all processors in its affinity are busy** executing higher-priority tasks

The Linux scheduler **never violates** a task's affinity

A **higher-priority process never migrates** to schedule a lower-priority process

Objective

Is the APA interface just an implementation detail or does it have interesting **theoretical implications**?

How can we derive schedulability **guarantees** for APA schedulers?

Does APA scheduling help **improve** schedulability?

Objective

Is the APA interface just an implementation detail or does it have interesting **theoretical implications**?

guarantees for APA schedulers?

Does APA scheduling help **improve** schedulability?

System model

- Sporadic task model with **arbitrary** deadlines

System model

- Sporadic task model with **arbitrary** deadlines
- Priority assignment
  - This talk and Linux: **fixed-priority** (FP)
  - In the paper: any **job-level fixed priority** (JLFP) e.g., earliest deadline first (EDF)

Is the APA interface just an implementation detail?

No.

APA scheduling **strictly dominates** global, clustered, and partitioned JLFP scheduling

Is the APA interface just an implementation detail?

No.

APA scheduling **strictly dominates** global, clustered, and partitioned JLFP scheduling

- APA scheduling is general (**dominance**)
- Workloads that are only schedulable under APA scheduling (and therefore, **strict** dominance)

Example

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

- Real-time workload
- Multiprocessor with **2 CPUs**

Example

Relative-deadline

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
[illegible]	[illegible]	[illegible]	,000
[illegible]	[illegible]	[illegible]	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

Worst-case execution time

Period

- Real-time workload
- Multiprocessor with **2 CPUs**

Example

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High density tasks

High utilization tasks

Partitioned scheduling (2 CPUs)?

$Task_i$	**C_i**	**D_i**	**P_i**
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High utilization tasks

- Partition tasks **T_5**, **T_6**, and **T_7**
- Utilizations of **T_5**, **T_6**, and **T_7** are **51%**, **50.1%**, and **50%**

Partitioned scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
T_6	501	1,000	1,000
T_7	500	1,000	1,000

- Partition tasks T_5, T_6, and T_7
- Utilizations of T_5, T_6, and T_7

High utilization tasks

The workload **cannot be partitioned** onto two CPUs

Global scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High density tasks

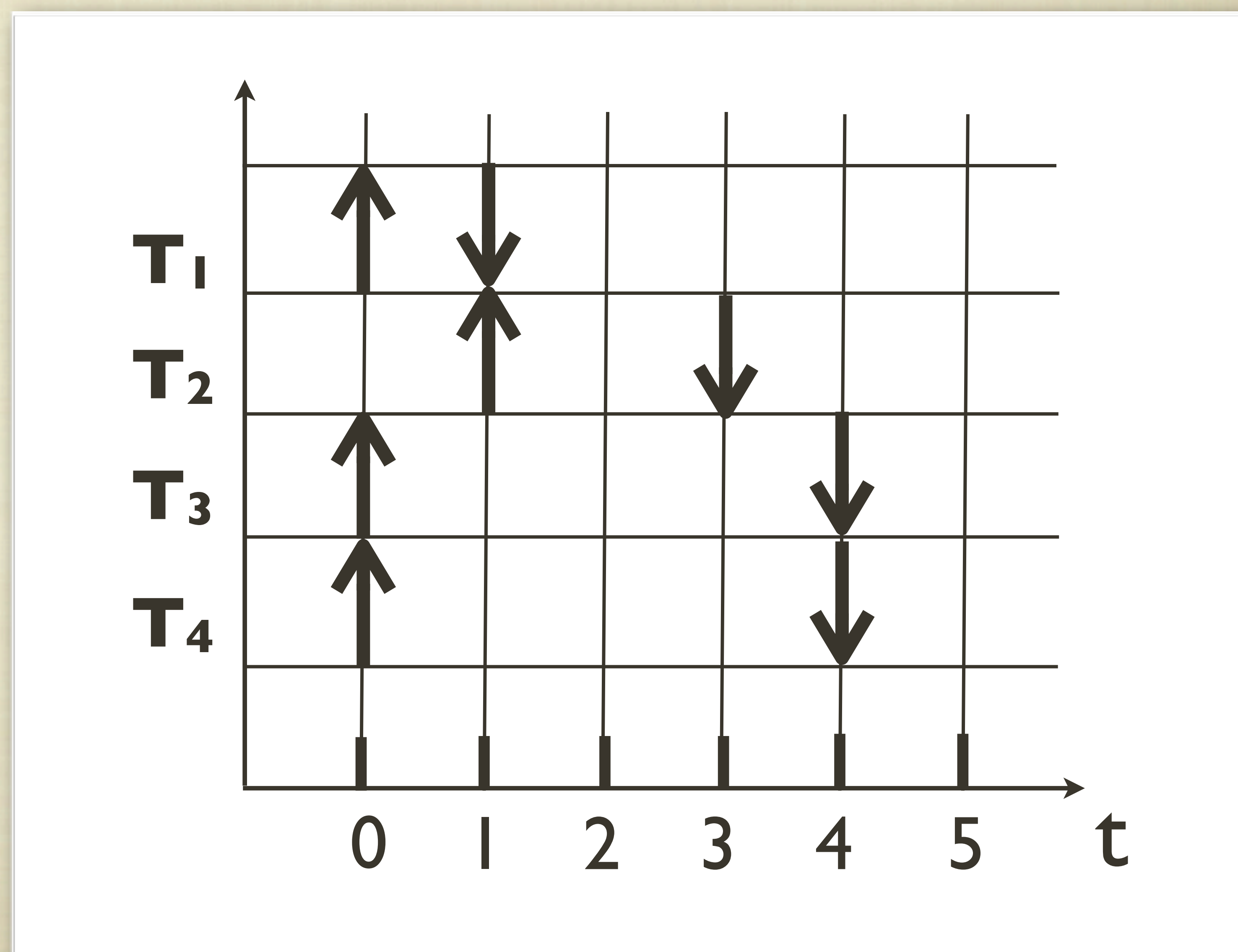
- FP: $T_1 > T_2 > T_3 > T_4$

Global scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High density tasks

- FP: $T_1 > T_2 > T_3 > T_4$

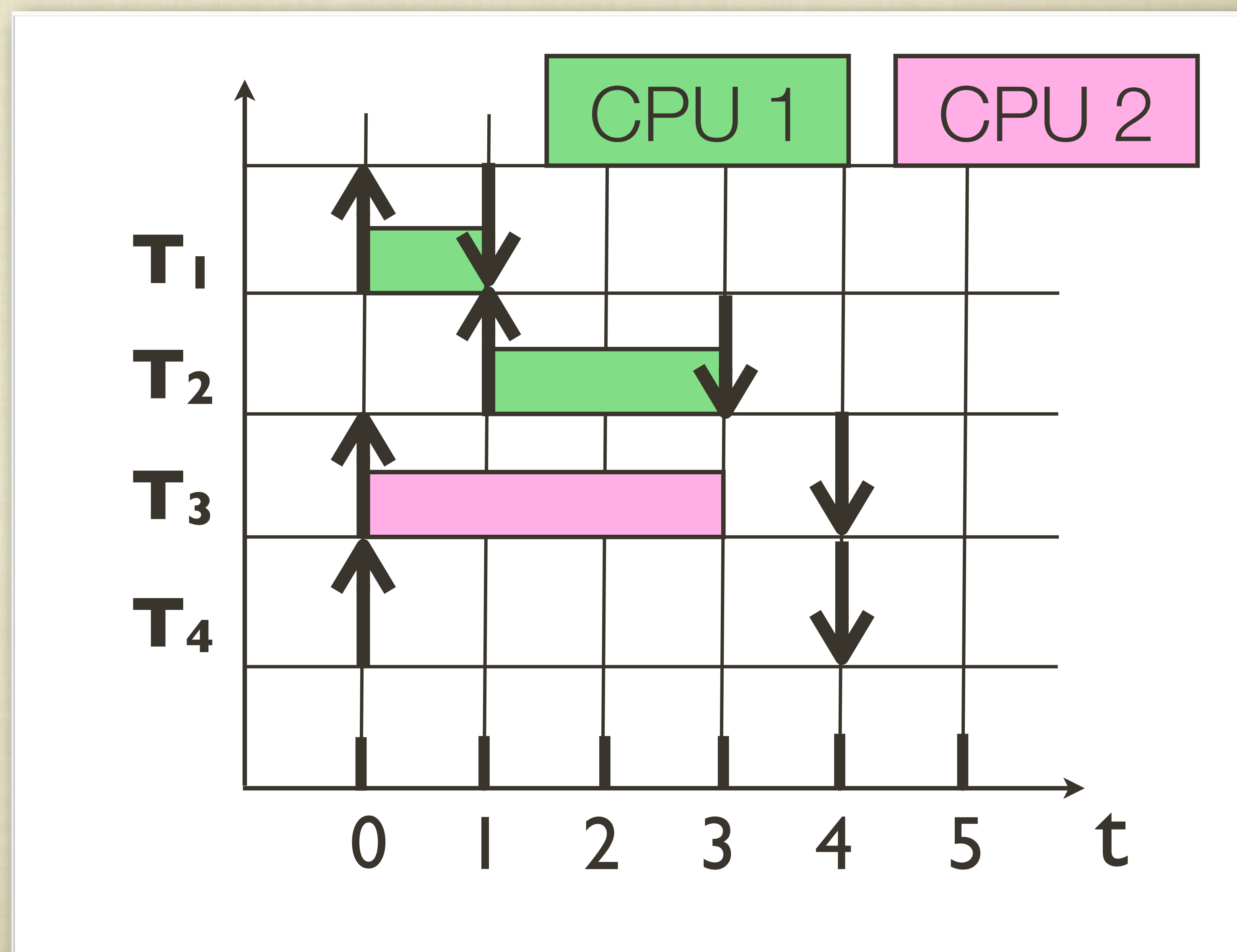


Global scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High density tasks

- FP: $T_1 > T_2 > T_3 > T_4$

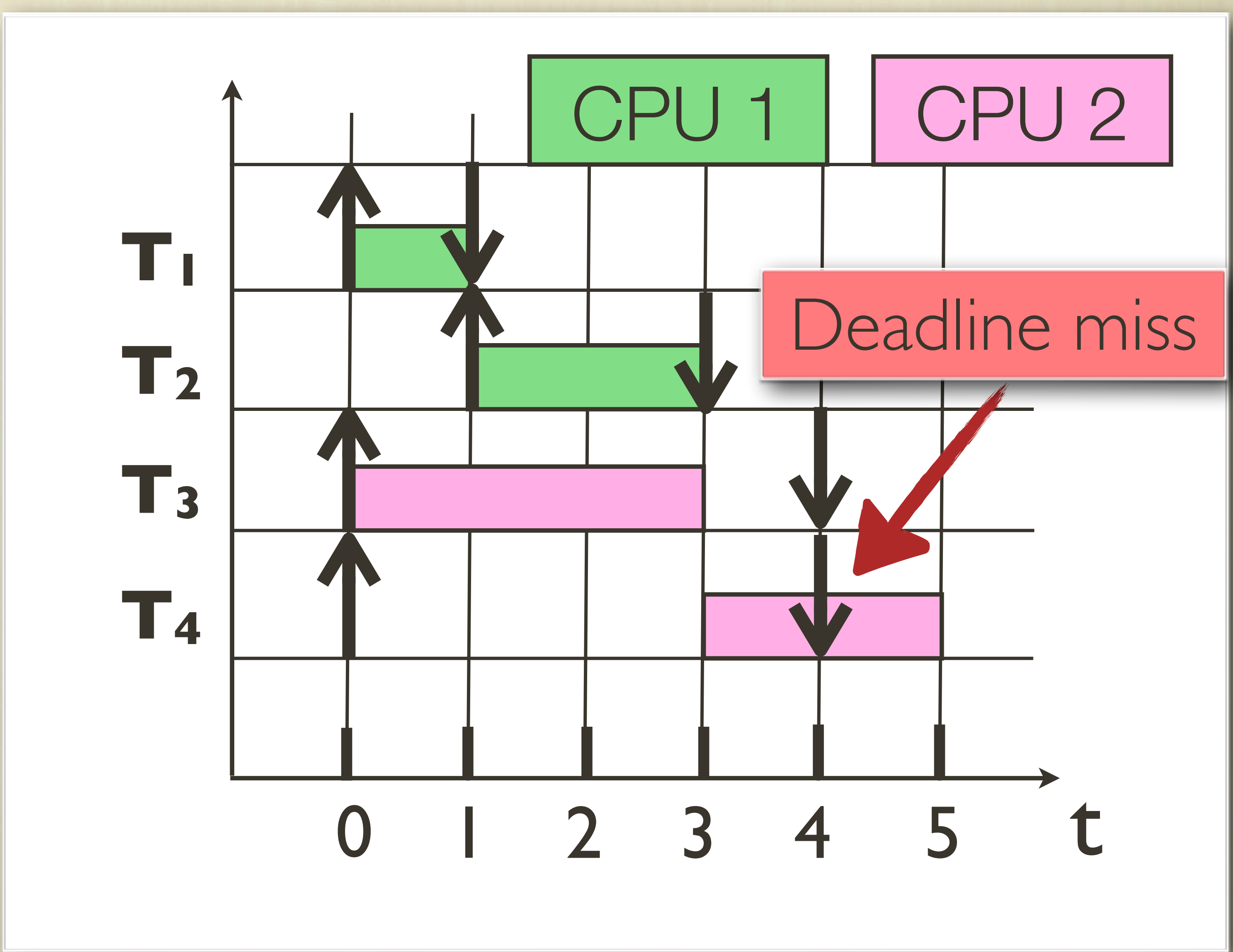


Global scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High density tasks

- FP: $T_1 > T_2 > T_3 > T_4$



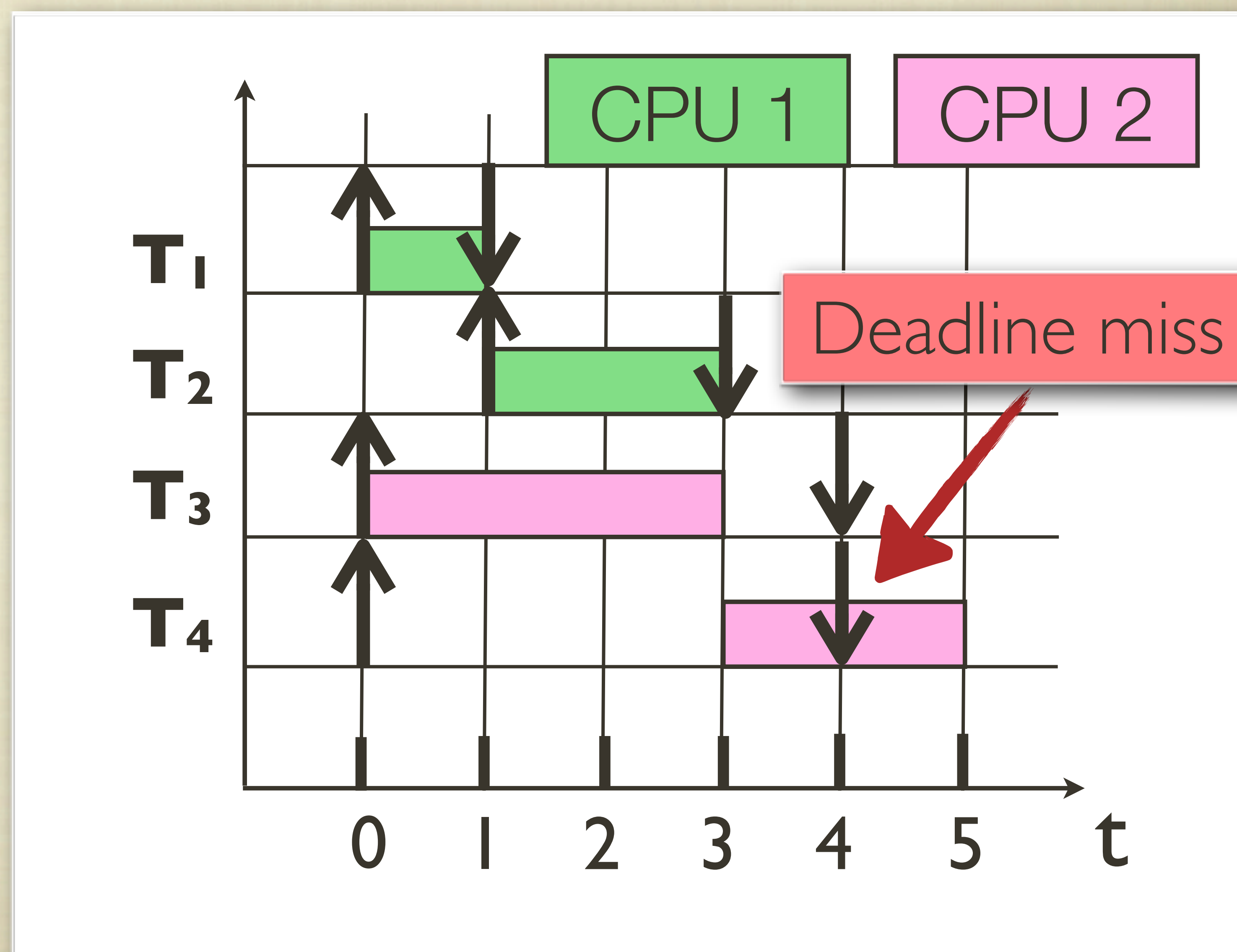
Global scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High density tasks

- FP: T

What if we switch the priority of $\mathbf{T_3}$ and $\mathbf{T_4}$?



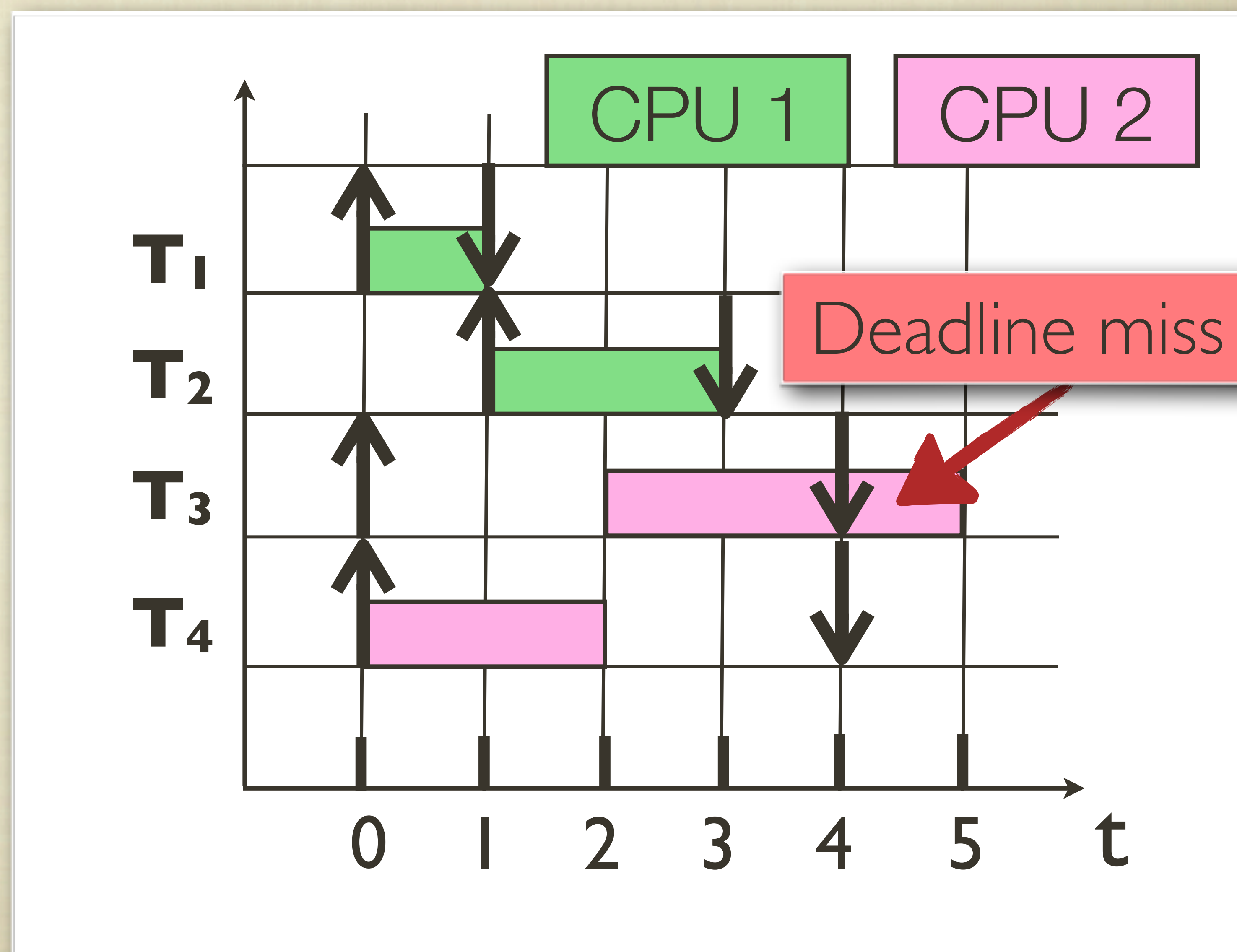
Global scheduling (2 CPUs)?

Task$_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High density tasks

What if we switch the priority of $\mathbf{T_3}$ and $\mathbf{T_4}$?

- FP: T

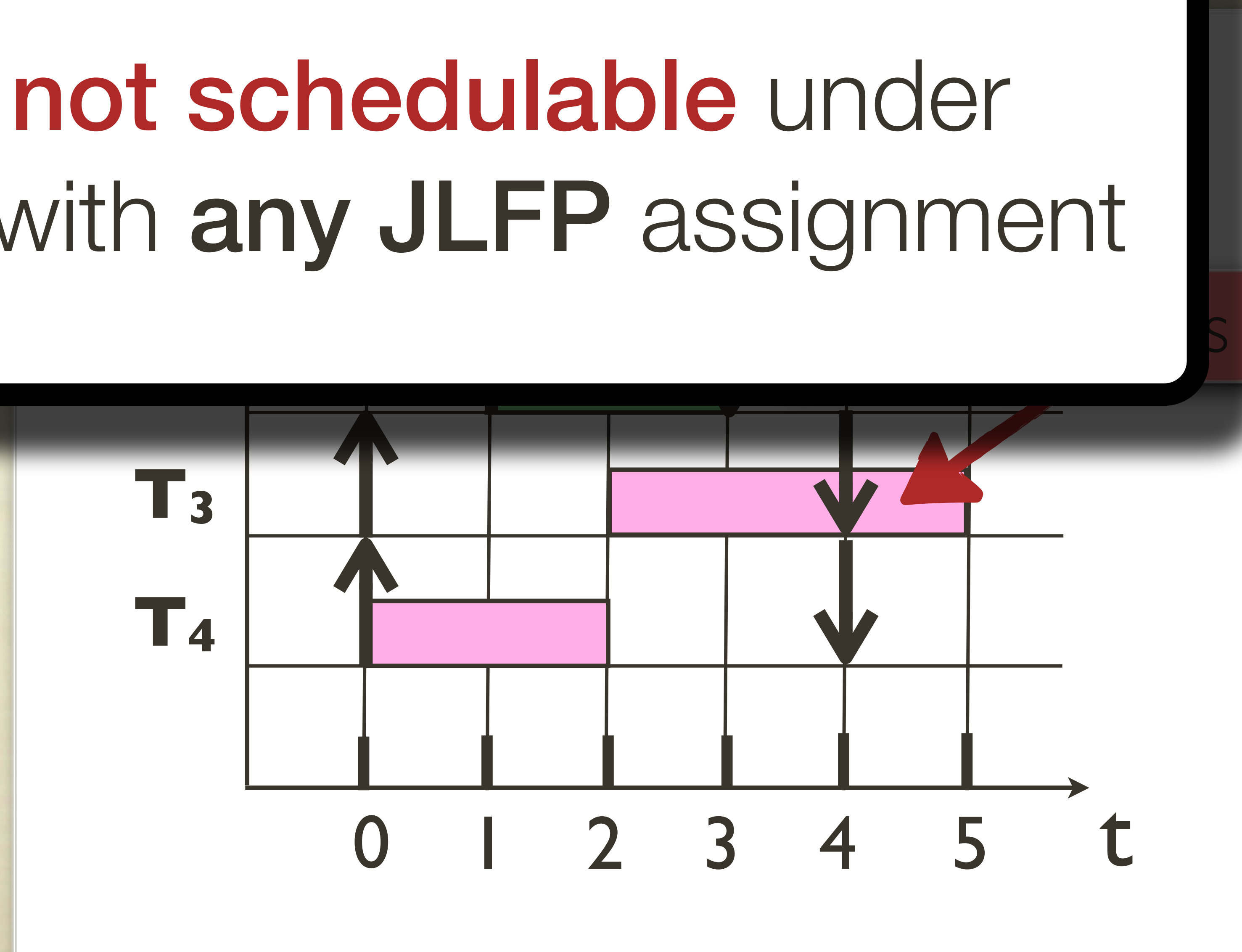


Global scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

High density tasks

- FP: $T_1 > T_2 > T_3 > T_4$



The workload is **not schedulable** under **global** scheduling with **any JLFP** assignment

APA scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

APA scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

Not schedulable under global scheduling, but can be **partitioned**

APA scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

Not schedulable under global scheduling, but can be **partitioned**

Any two out of these three tasks can be partitioned

APA scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

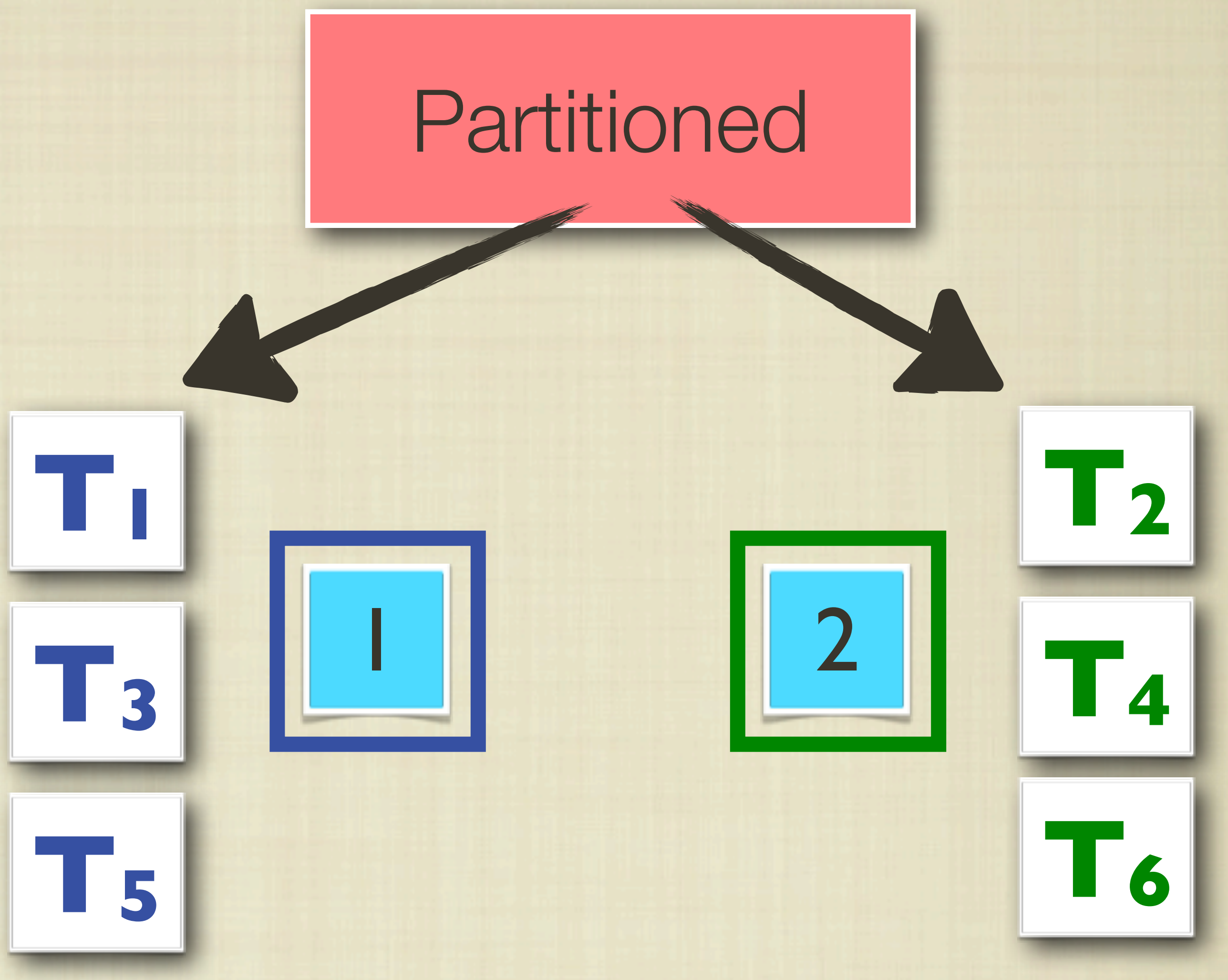
Not schedulable under global scheduling, but can be **partitioned**

Partition tasks $\mathbf{T_1}$-$\mathbf{T_6}$, and assign $\mathbf{T_7}$ global-like affinity

Any two out of these three tasks can be partitioned

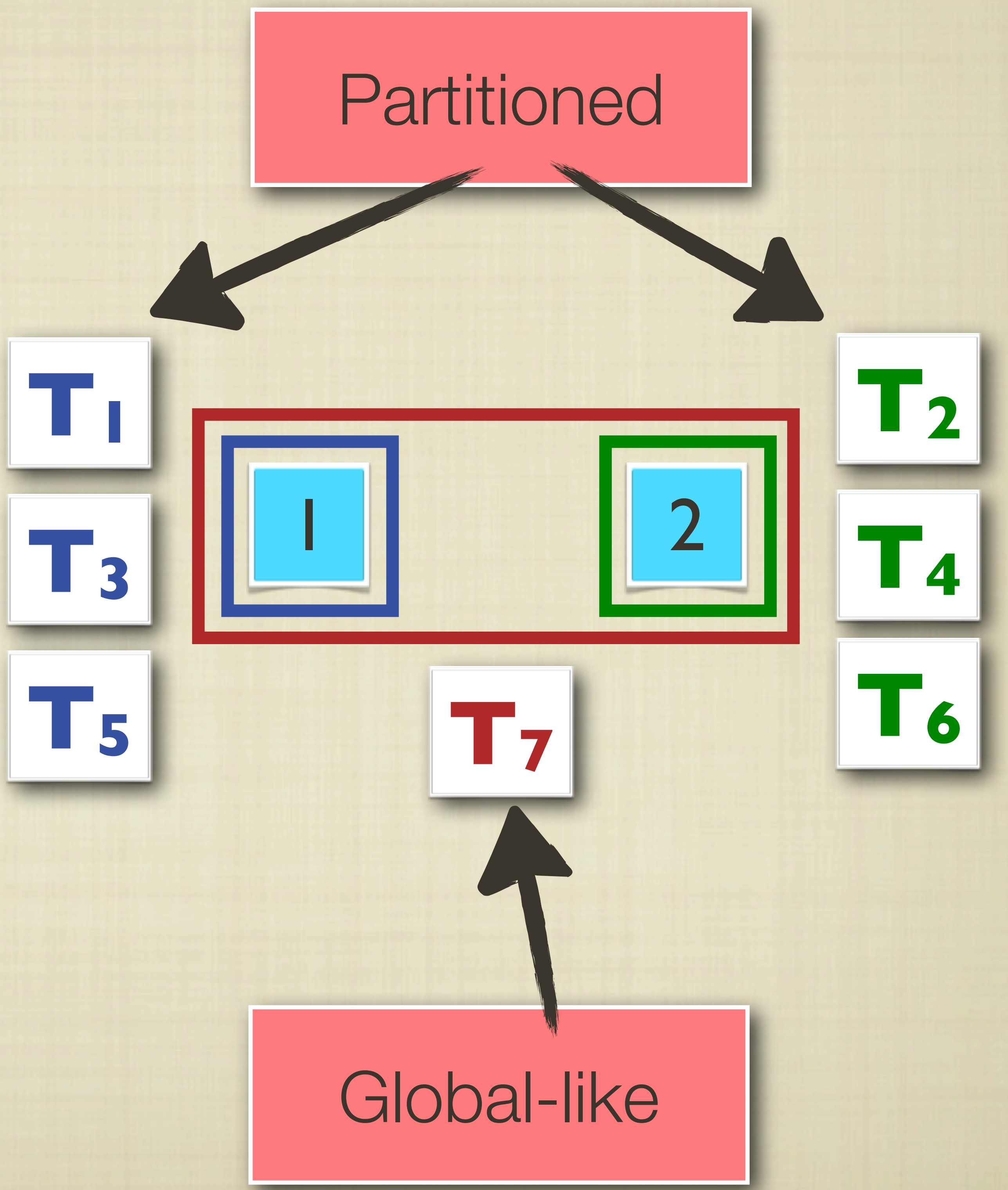
APA scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000



APA scheduling (2 CPUs)?

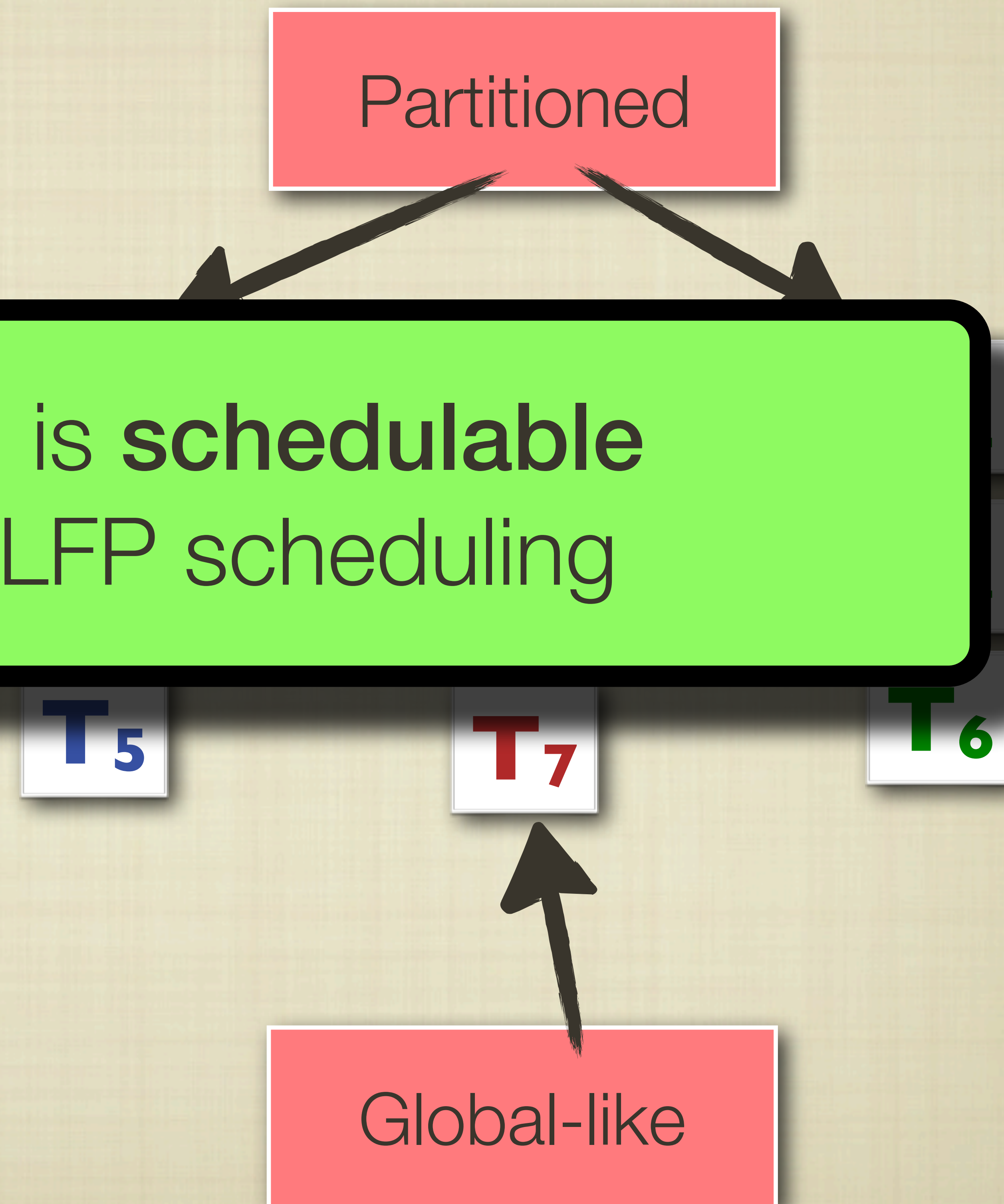
$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_2	2	2	1,000
T_3	3	4	1,000
T_4	2	4	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000



APA scheduling (2 CPUs)?

$Task_i$	C_i	D_i	P_i
T_1	1	1	1,000
T_5	51	100	100
T_6	501	1,000	1,000
T_7	500	1,000	1,000

The workload is **schedulable**
under **APA** JLFP scheduling



Objective

✓

APA scheduling **strictly dominates** global, clustered, and partitioned JLFP scheduling.

How can we derive schedulability **guarantees** for APA schedulers?

Does APA scheduling help **improve** schedulability?

Objective

✓ APA scheduling **strictly dominates** global,

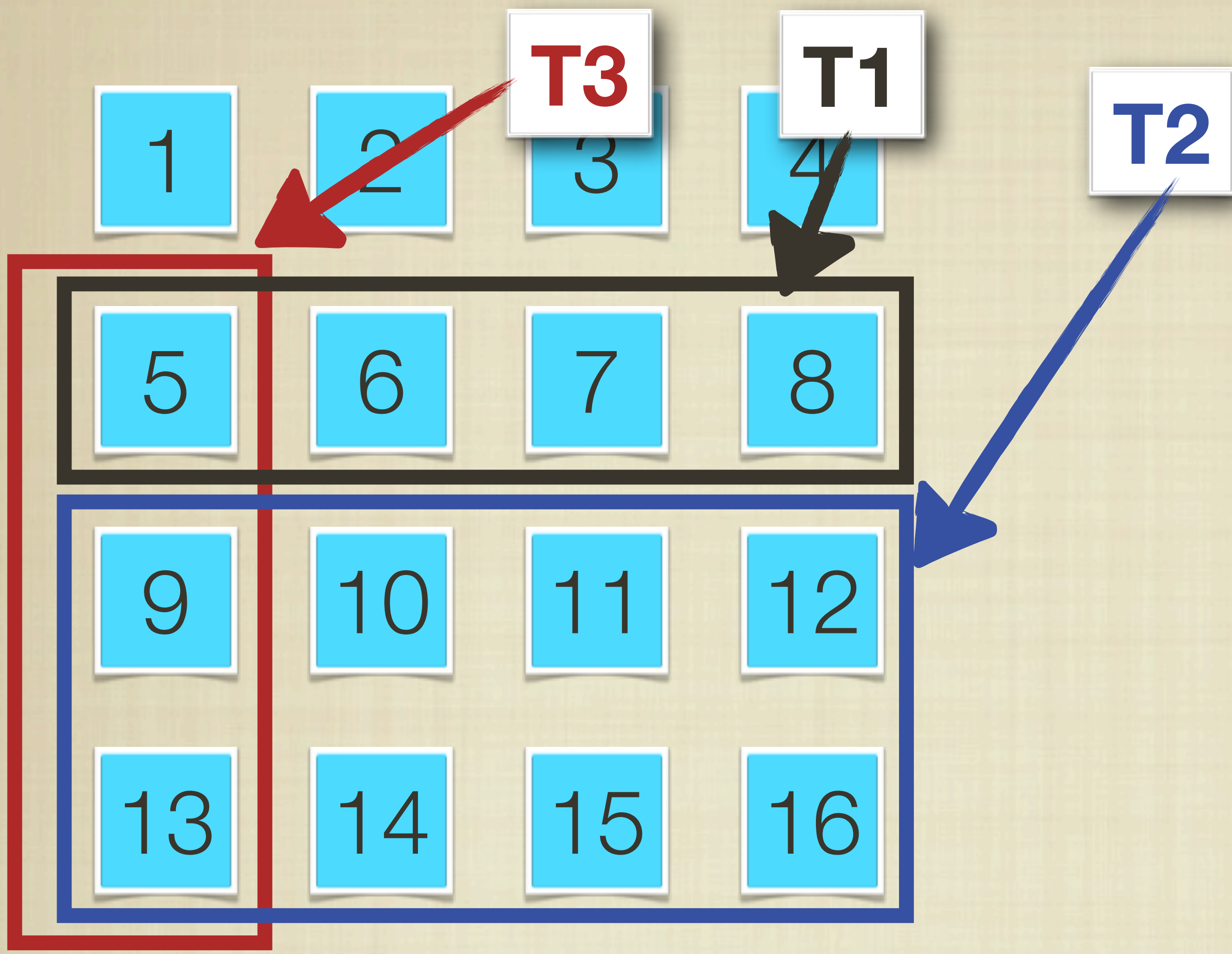
How can we derive schedulability **guarantees** for APA schedulers?

Does APA scheduling help **improve** schedulability?

Schedulability Analysis (simple)

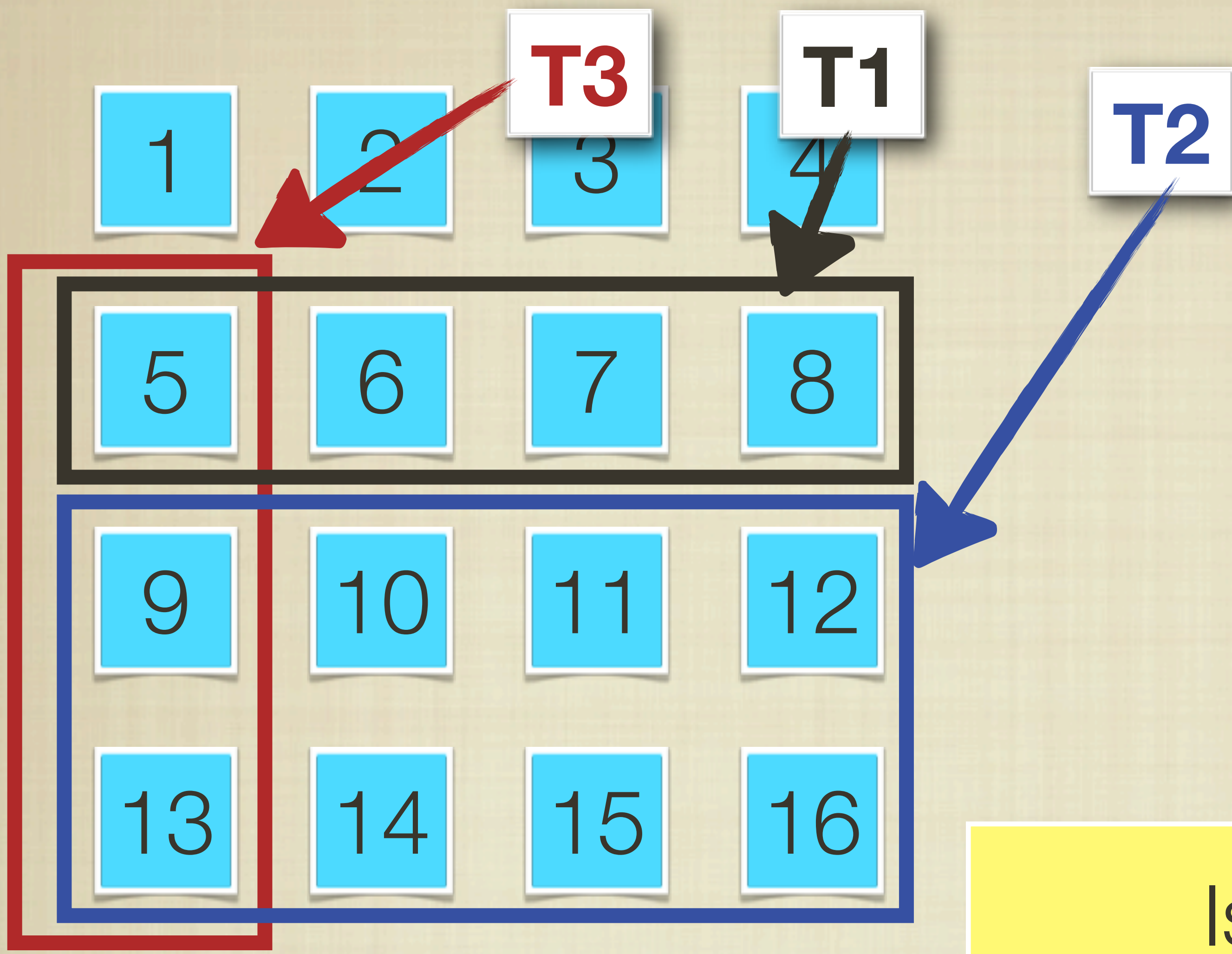
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Schedulability Analysis (simple)



FP: $\mathbf{T_1} > \mathbf{T_2} > \mathbf{T_3}$

Schedulability Analysis (simple)

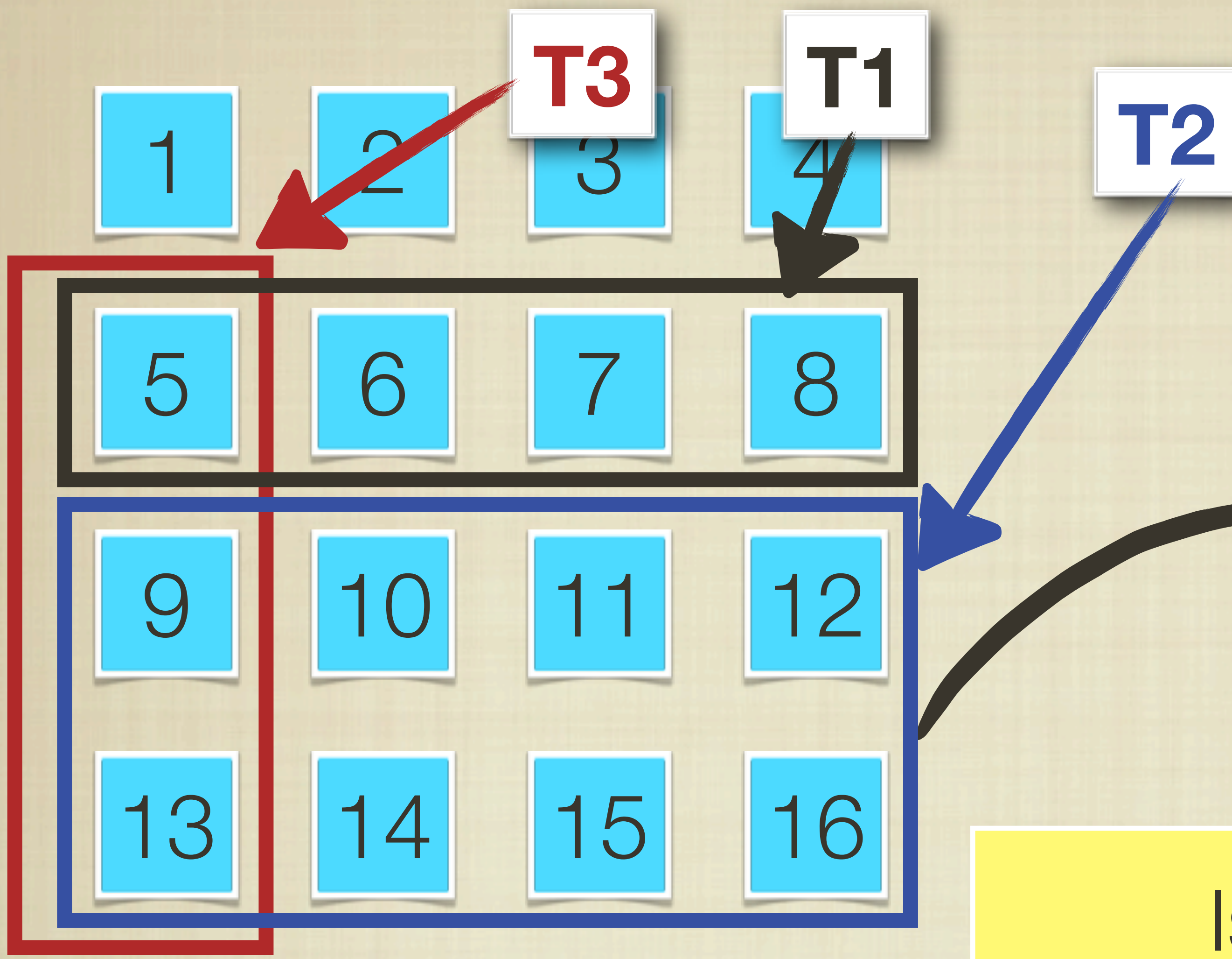


FP: $\mathbf{T_1} > \mathbf{T_2} > \mathbf{T_3}$

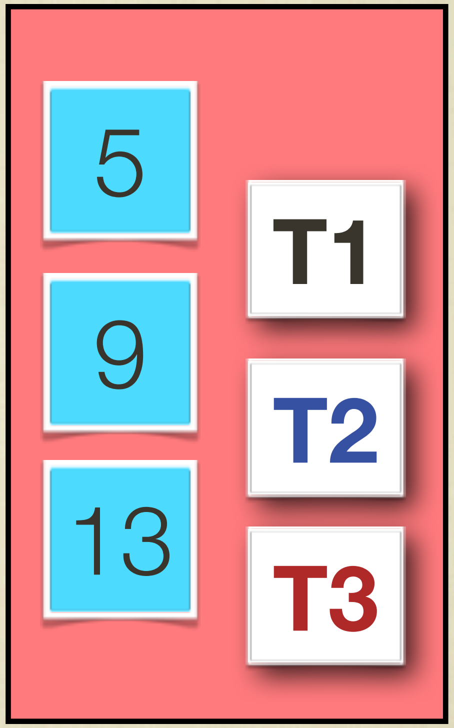
Is $\mathbf{T_3}$ schedulable?

Schedulability Analysis (simple)

FP: $T_1 > T_2 > T_3$

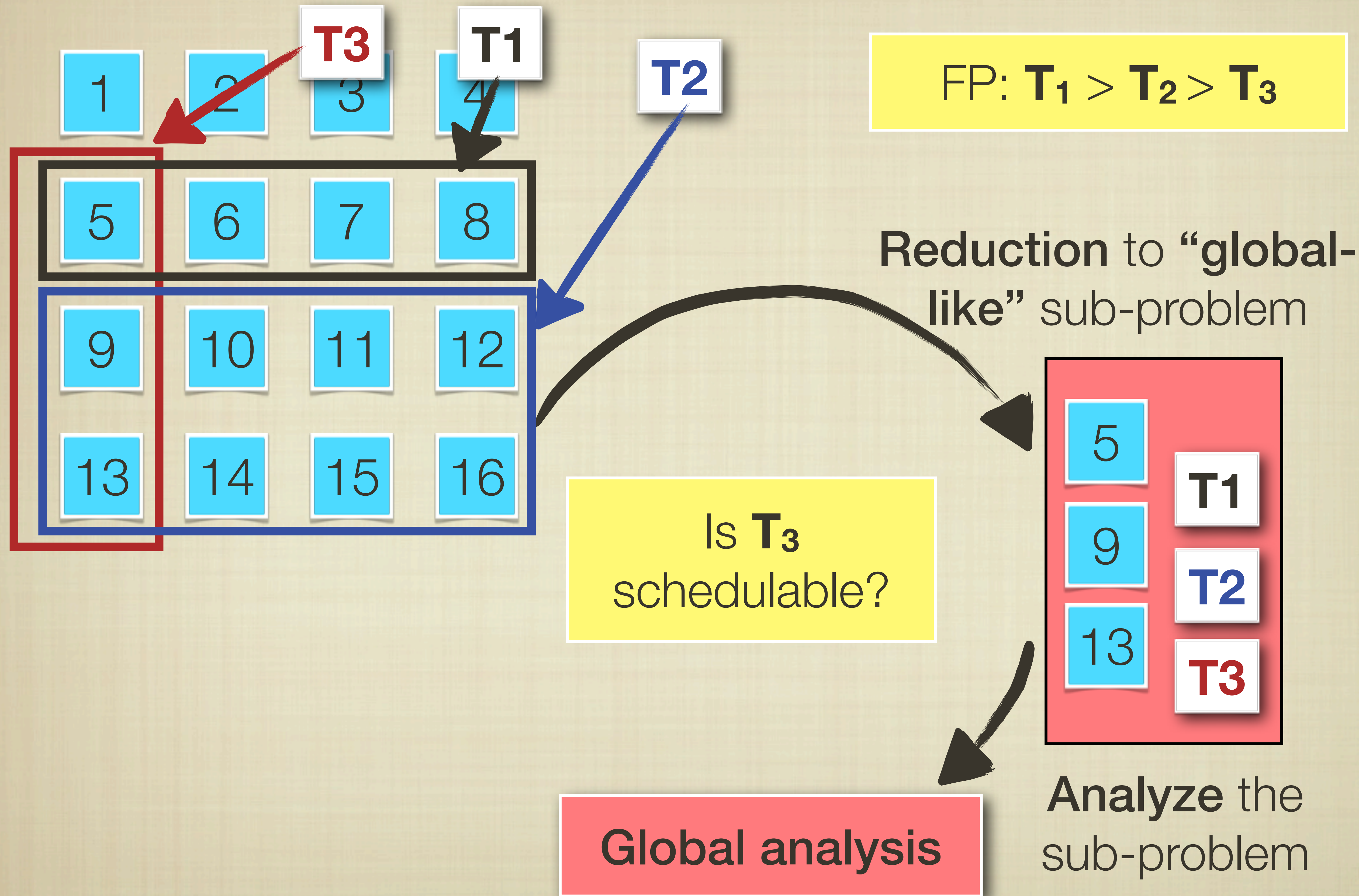


Reduction to "**global-like**" sub-problem

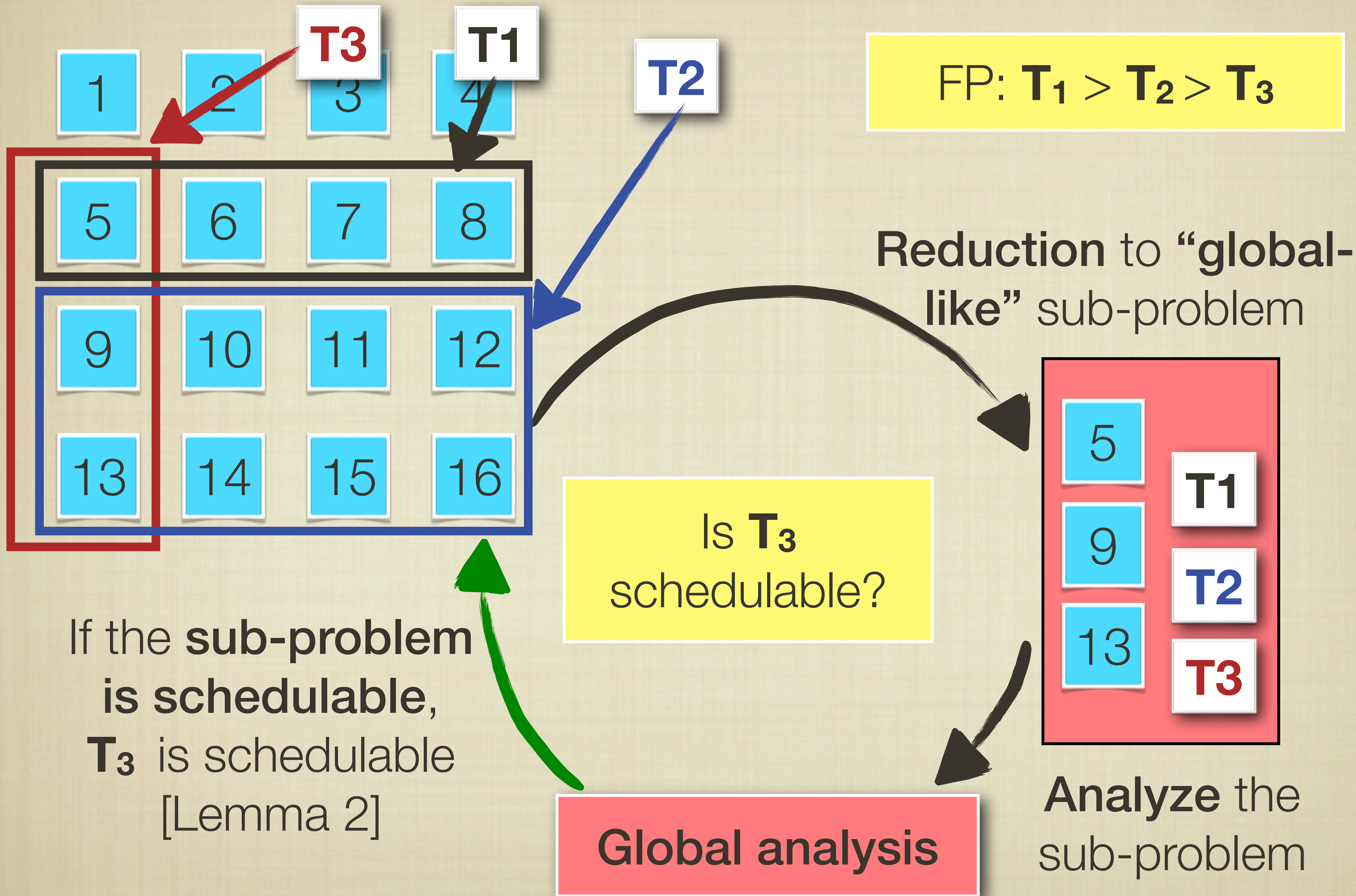


Is $\mathbf{T_3}$ schedulable?

Schedulability Analysis (simple)



Schedulability Analysis (simple)



Schedulability Analysis (simple)

- Why does the approach work?

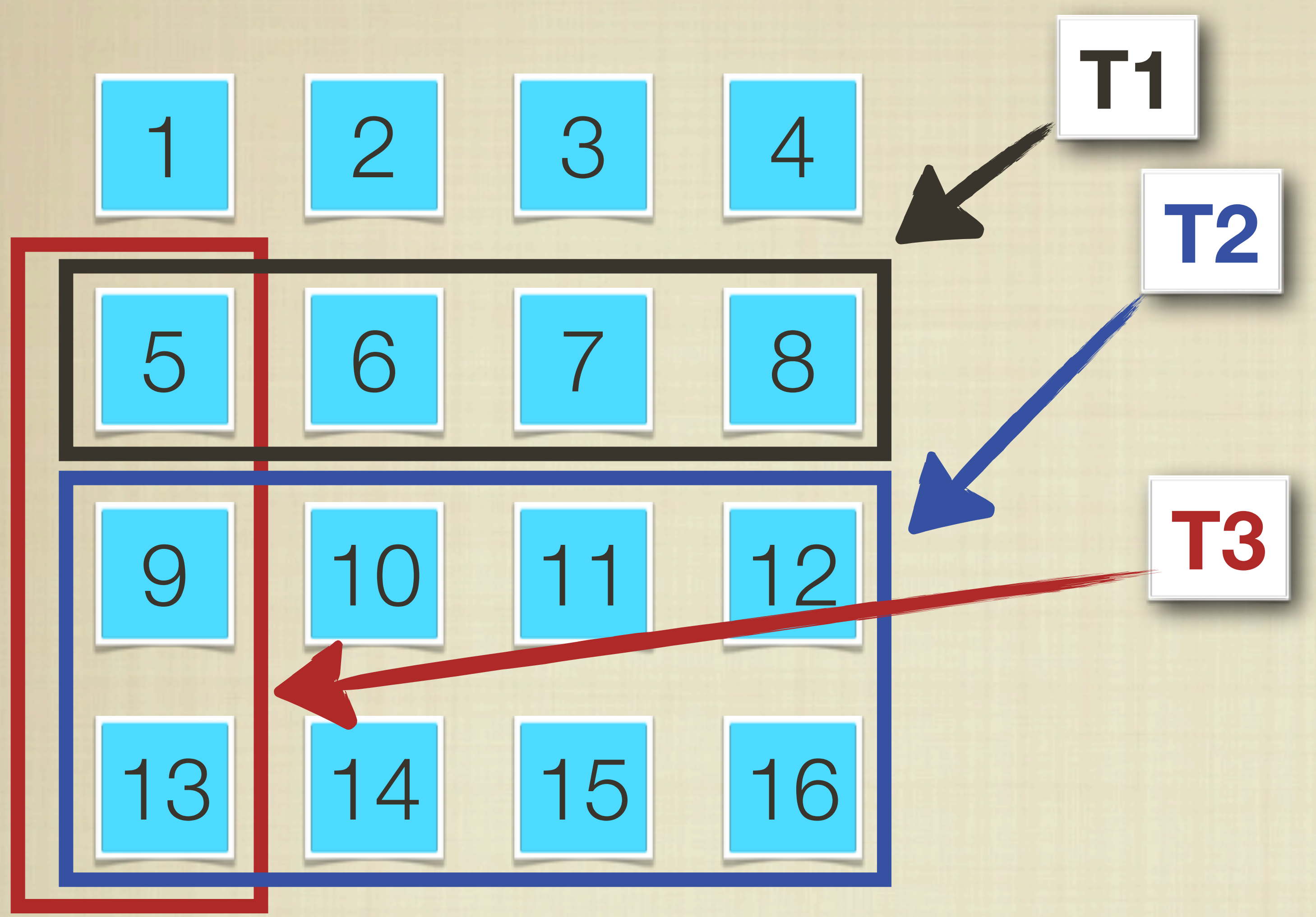
Reduction to **"global-like"** sub-problems

Considers the **worst-case** scenario

All **potentially interfering** tasks included

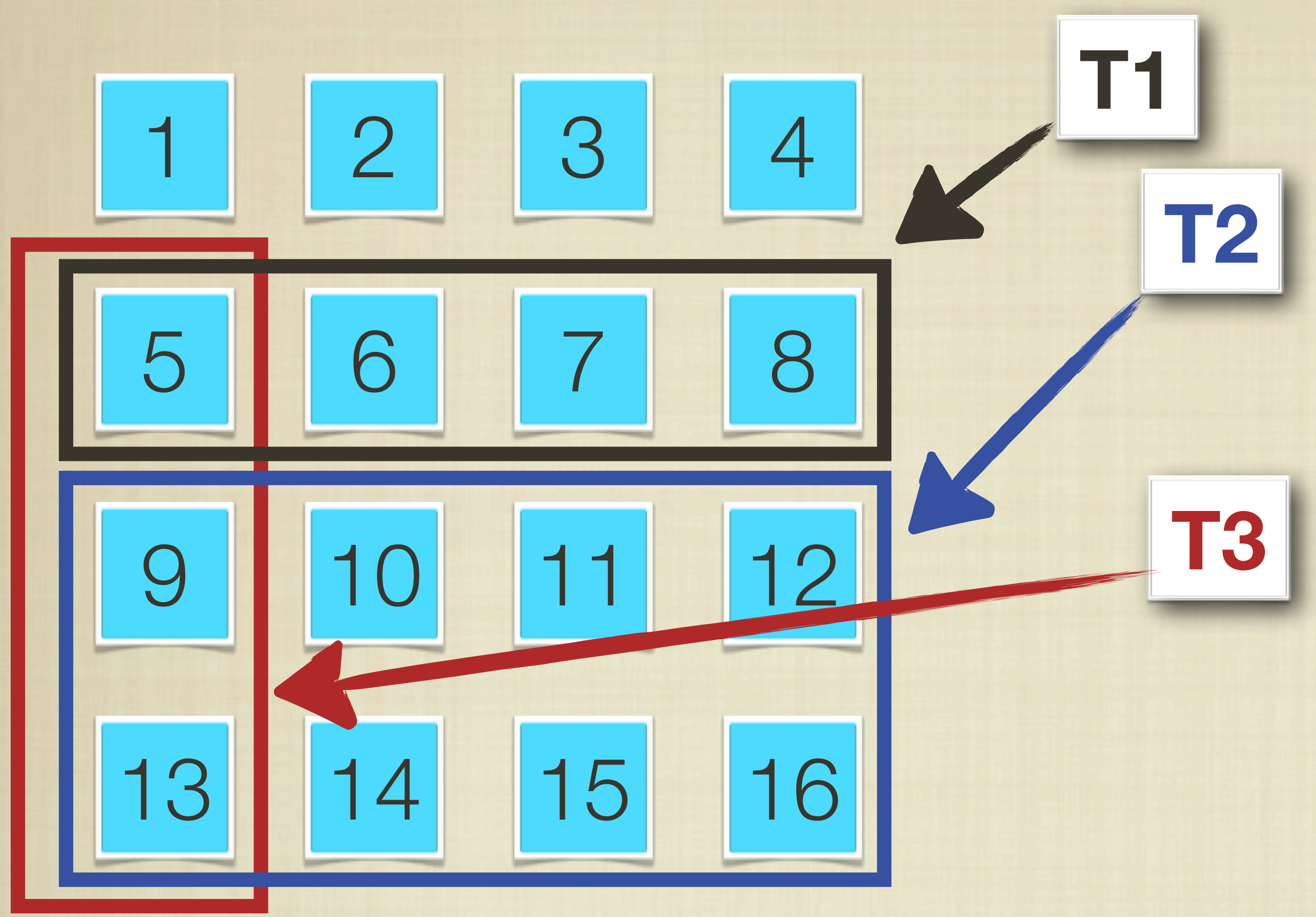
Pessimism?

Schedulability Analysis (exhaustive)



FP: $T_1 > T_2 > T_3$

Schedulability Analysis (exhaustive)



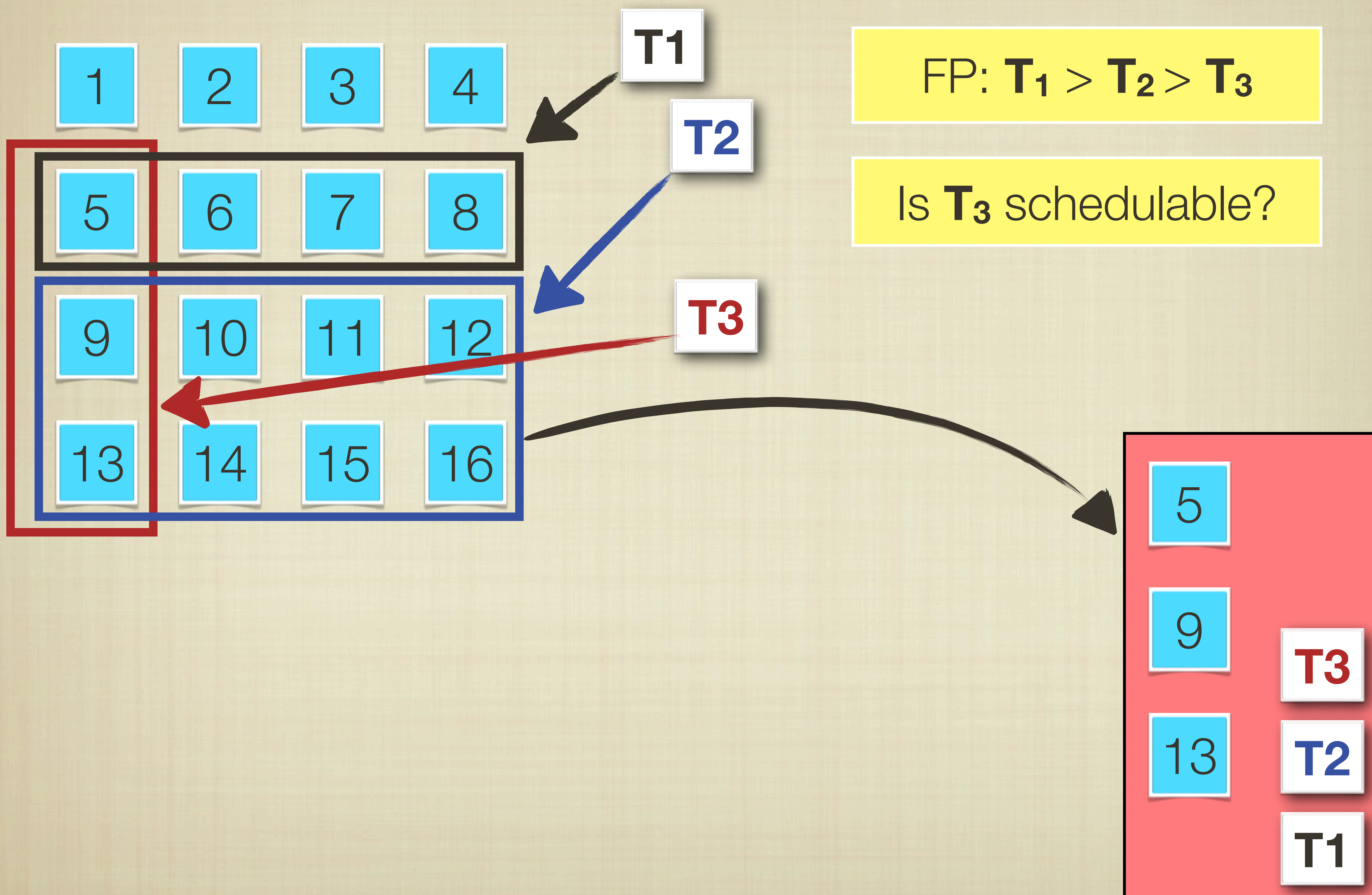
FP: $\mathbf{T_1} > \mathbf{T_2} > \mathbf{T_3}$

Is $\mathbf{T_3}$ schedulable?

Schedulability Analysis (exhaustive)

FP: $T_1 > T_2 > T_3$

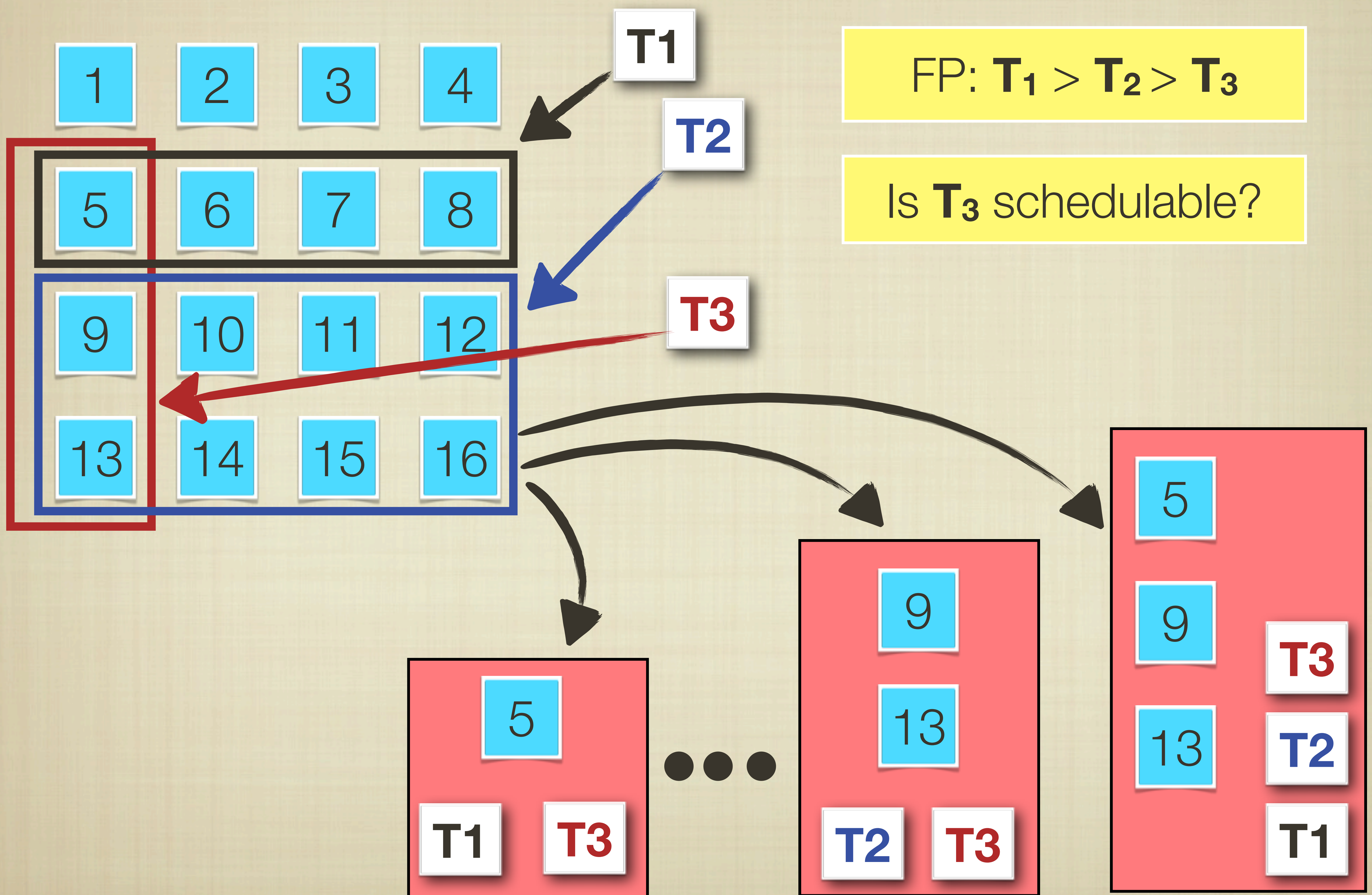
Is T_3 schedulable?



Schedulability Analysis (exhaustive)

FP: $T_1 > T_2 > T_3$

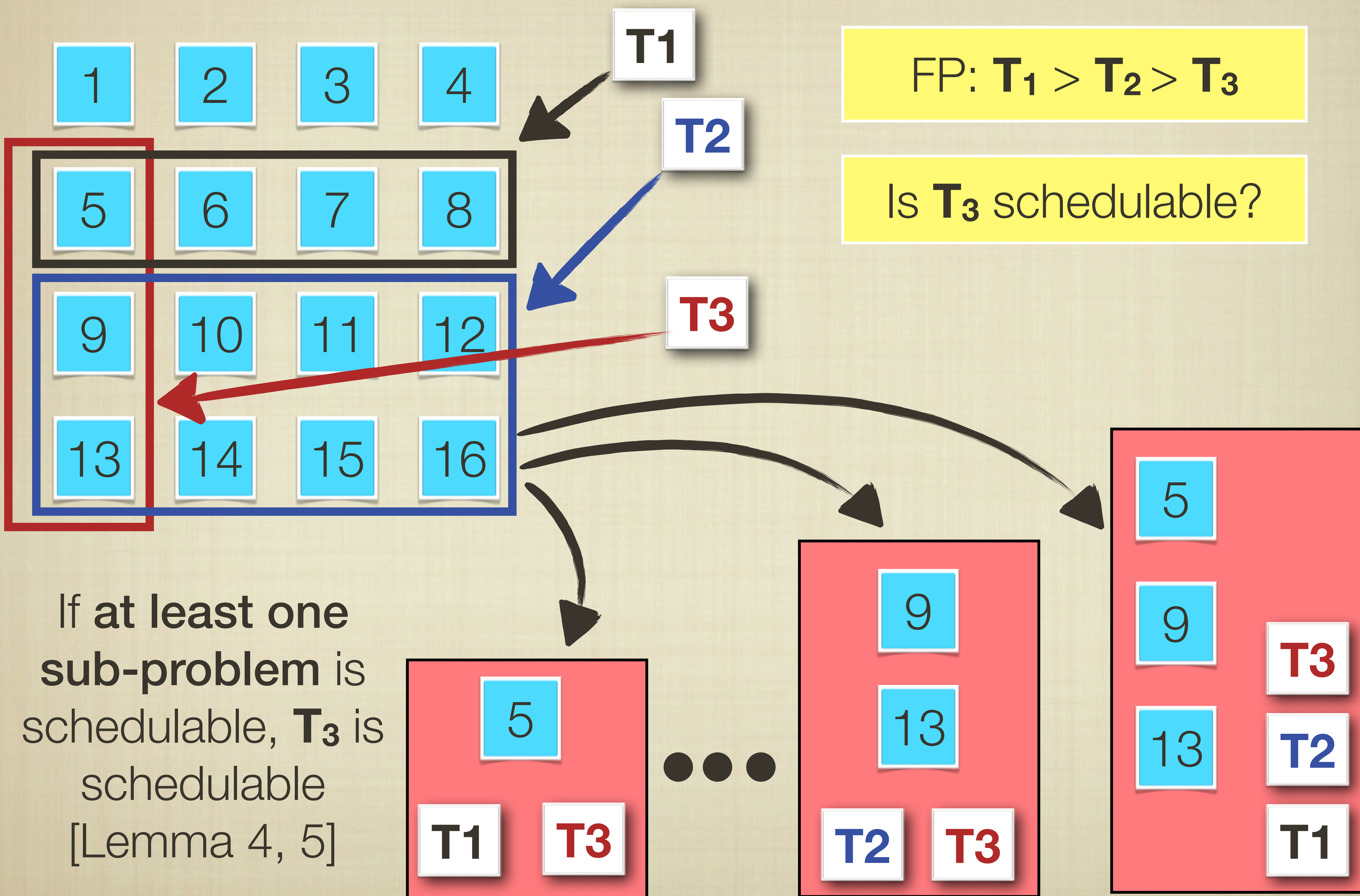
Is T_3 schedulable?



Schedulability Analysis (exhaustive)

FP: $\mathbf{T_1} > \mathbf{T_2} > \mathbf{T_3}$

Is $\mathbf{T_3}$ schedulable?



If **at least one sub-problem** is schedulable, $\mathbf{T_3}$ is schedulable [Lemma 4, 5]

Schedulability Analysis **(exhaustive)**

Problem? Number of sub-problems grows **exponentially**

Works only for multiprocessors with up to **8 CPUs**

Schedulability Analysis (heuristic-based)

- Need a **pruning** strategy
- For a task-affinity of size K, analyze **at most K subproblems** per task, not 2^K

Schedulability Analysis (heuristic-based)

- Need a **pruning** strategy
- For a task-affinity of size K, analyze **at most K subproblems** per task, not 2^K

```
while (not schedulable AND affinity is not empty)
  identify CPU that contributes most interference
  remove this CPU from affinity
  re-test with shrunk affinity
```

Isn't the reduction approach inherently pessimistic?

Isn’t the reduction approach inherently pessimistic?

- Analysis **limited** by **Linux scheduler** design

A **higher-priority process never migrates**
to schedule a lower-priority process

- “global-like” worst-case scenarios **possible**

Objective

✓

APA scheduling **strictly dominates** global, clustered, and partitioned JLFP scheduling.

We can derive schedulability guarantees for APA schedulers by **reduction** to global subproblems (can reuse **any global analysis**).

✓

Does APA scheduling help **improve** schedulability?

Objective

APA scheduling **strictly dominates** global,

Does APA scheduling help **improve** schedulability?

schedulers by **reduction** to global subproblems
(can reuse **any global analysis**).

Evaluation

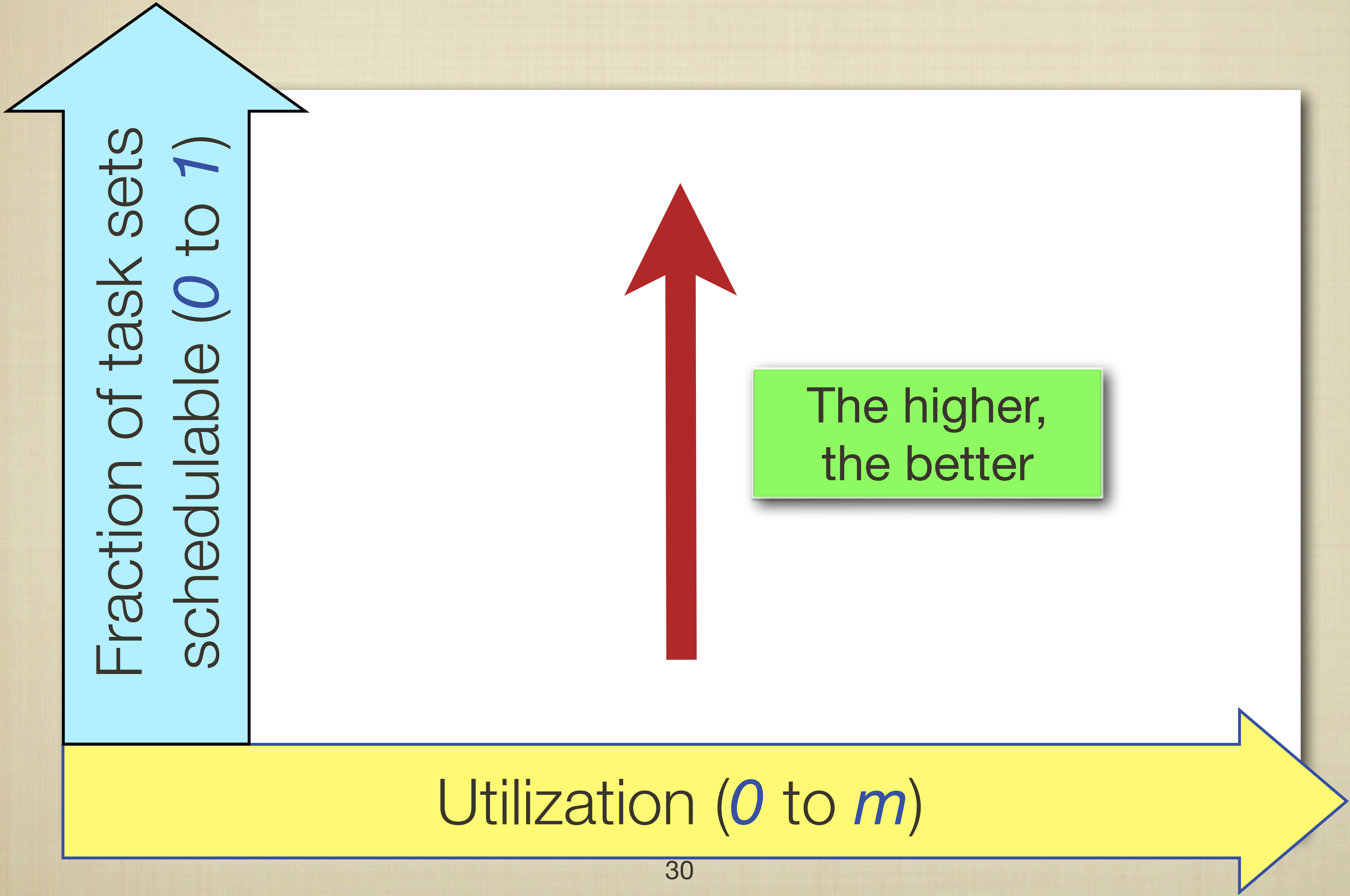
- Two sets of experiments:
  - Exhaustive vs. heuristic-based analysis
  - Global vs. partitioned vs. APA scheduling

Evaluation

- Emberson et al. task set generator [1]
  (task sets with **implicit** deadlines)
- Log-uniform distribution of periods [10ms,100ms]
- Number of CPUs (*m*) varied from 3 to 8
- Number of tasks ranging from *m+1* to *2.5m*

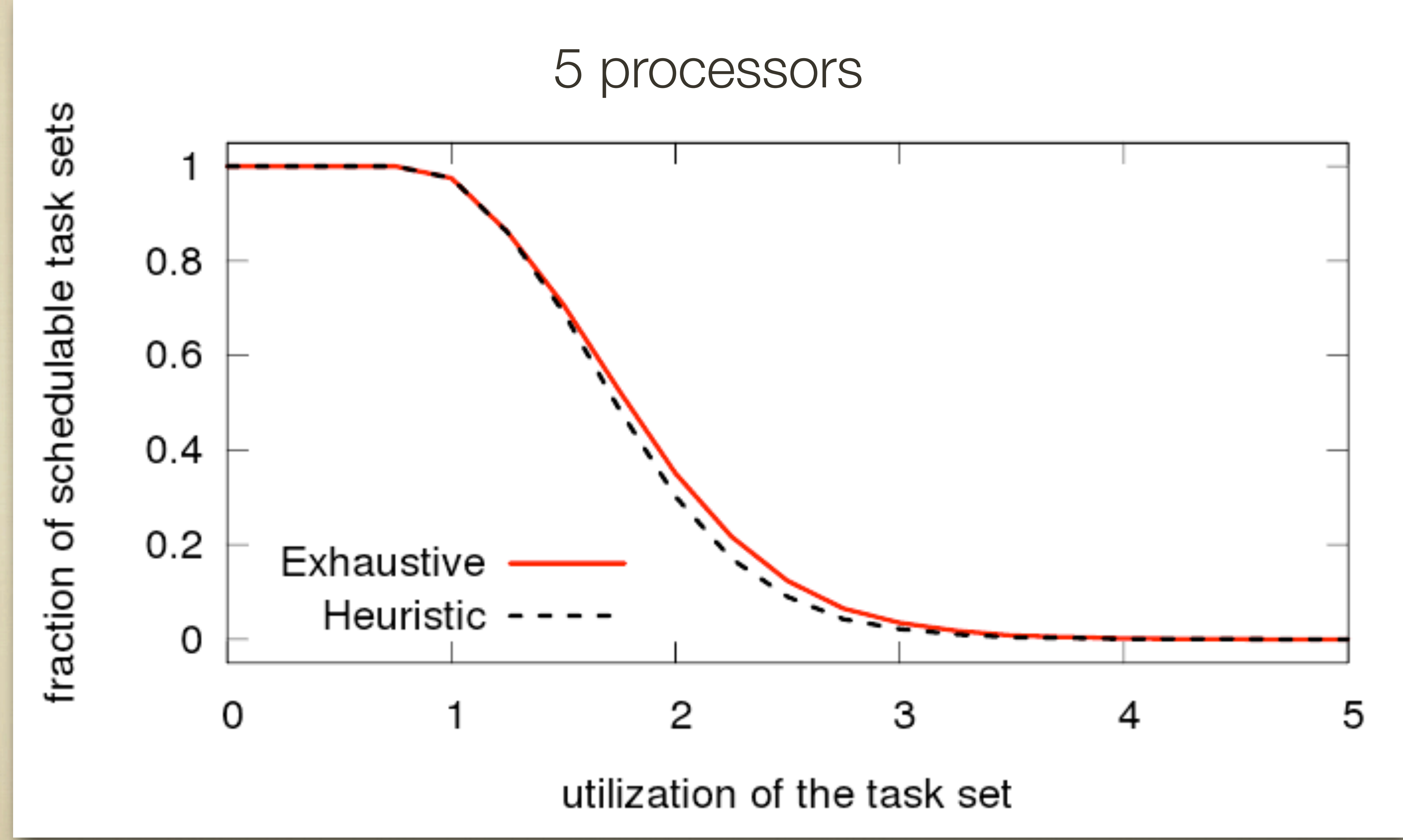
[1] P. Emberson, R. Stafford, and R. Davis, "Techniques for the synthesis of multiprocessor tasksets," 1st Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems, 2010.

Schedulability experiment graph

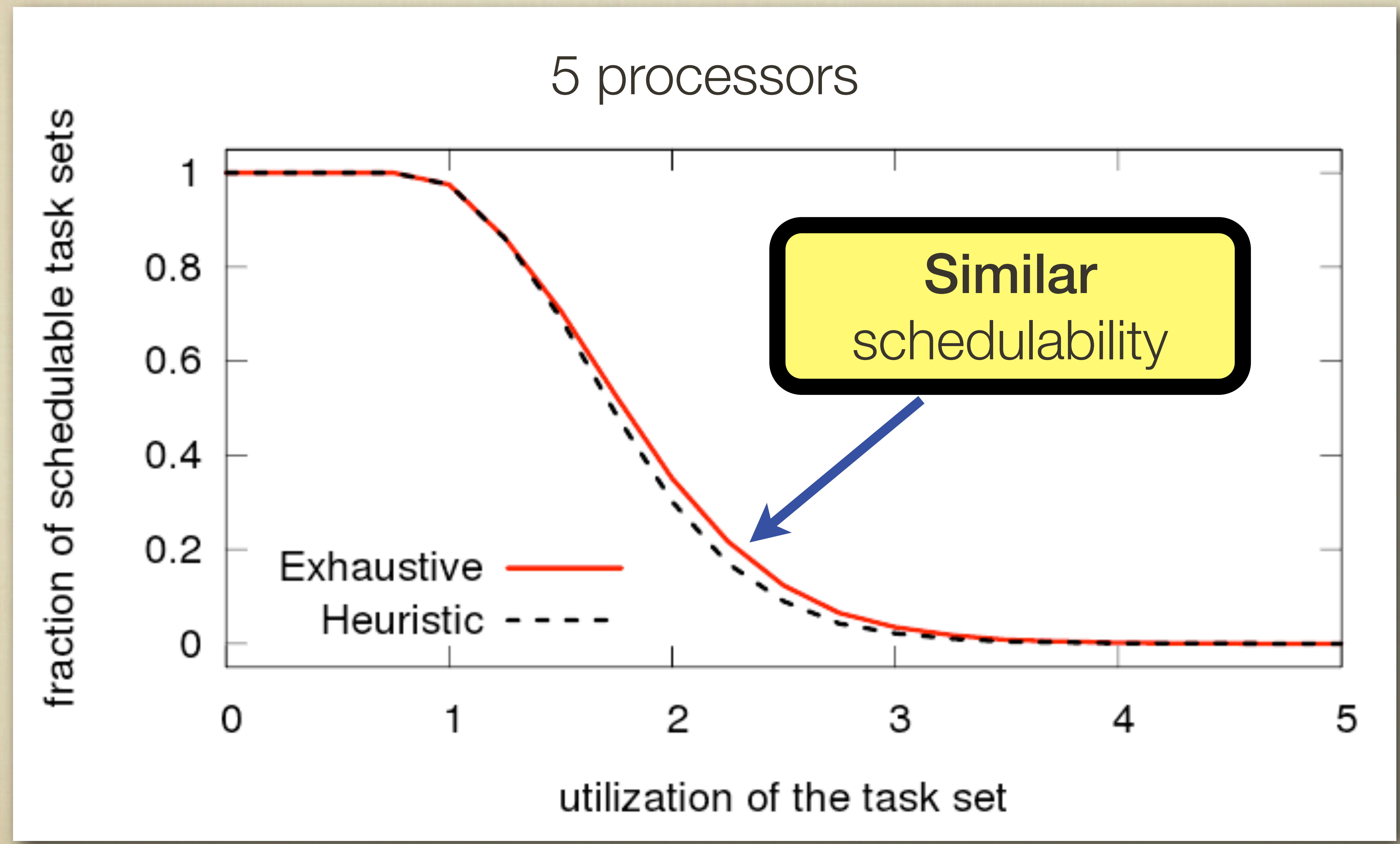


Experiment 1: exhaustive vs. heuristic-based analysis

Experiment 1: exhaustive vs. heuristic-based analysis

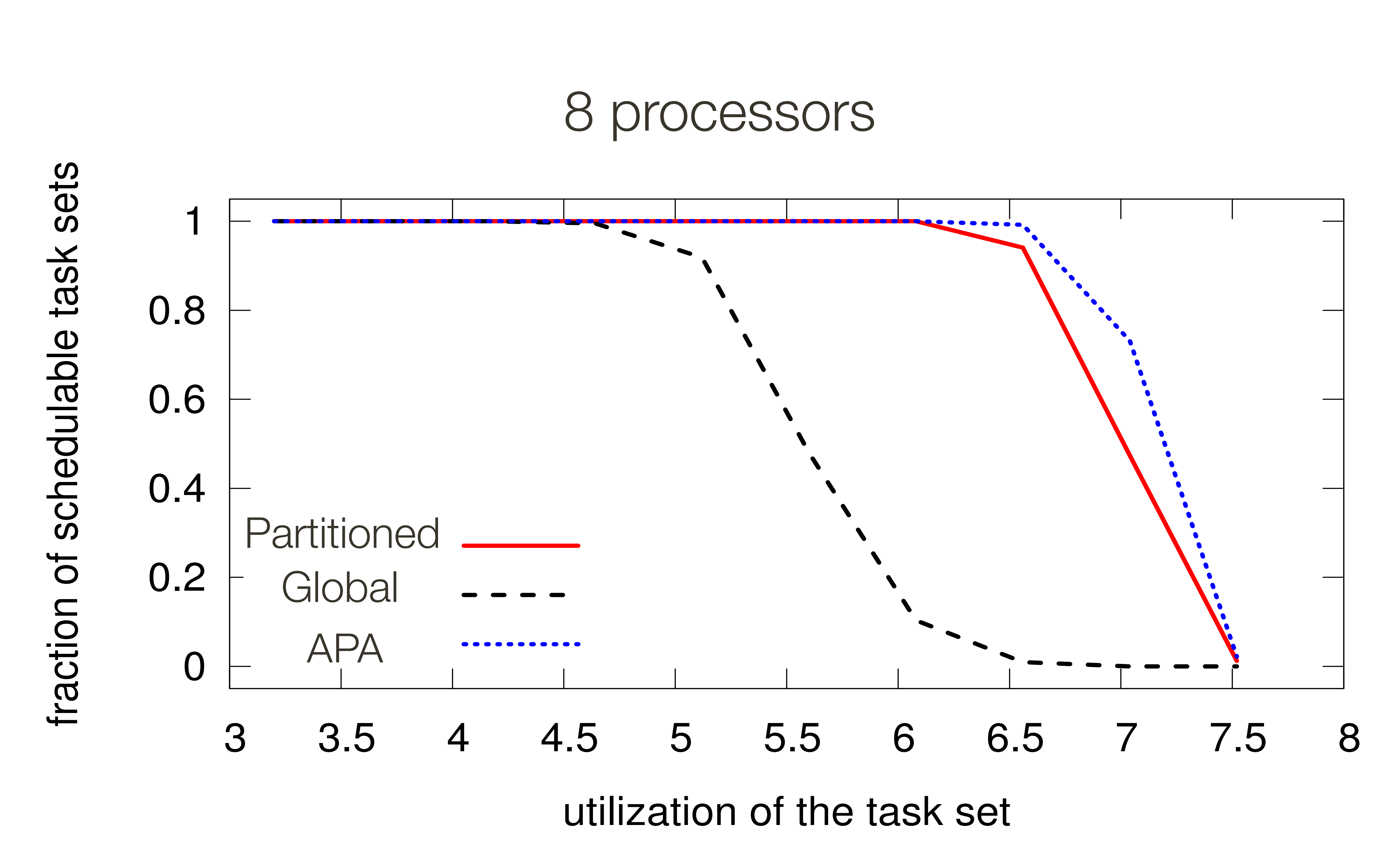


Experiment 1: exhaustive vs. heuristic-based analysis



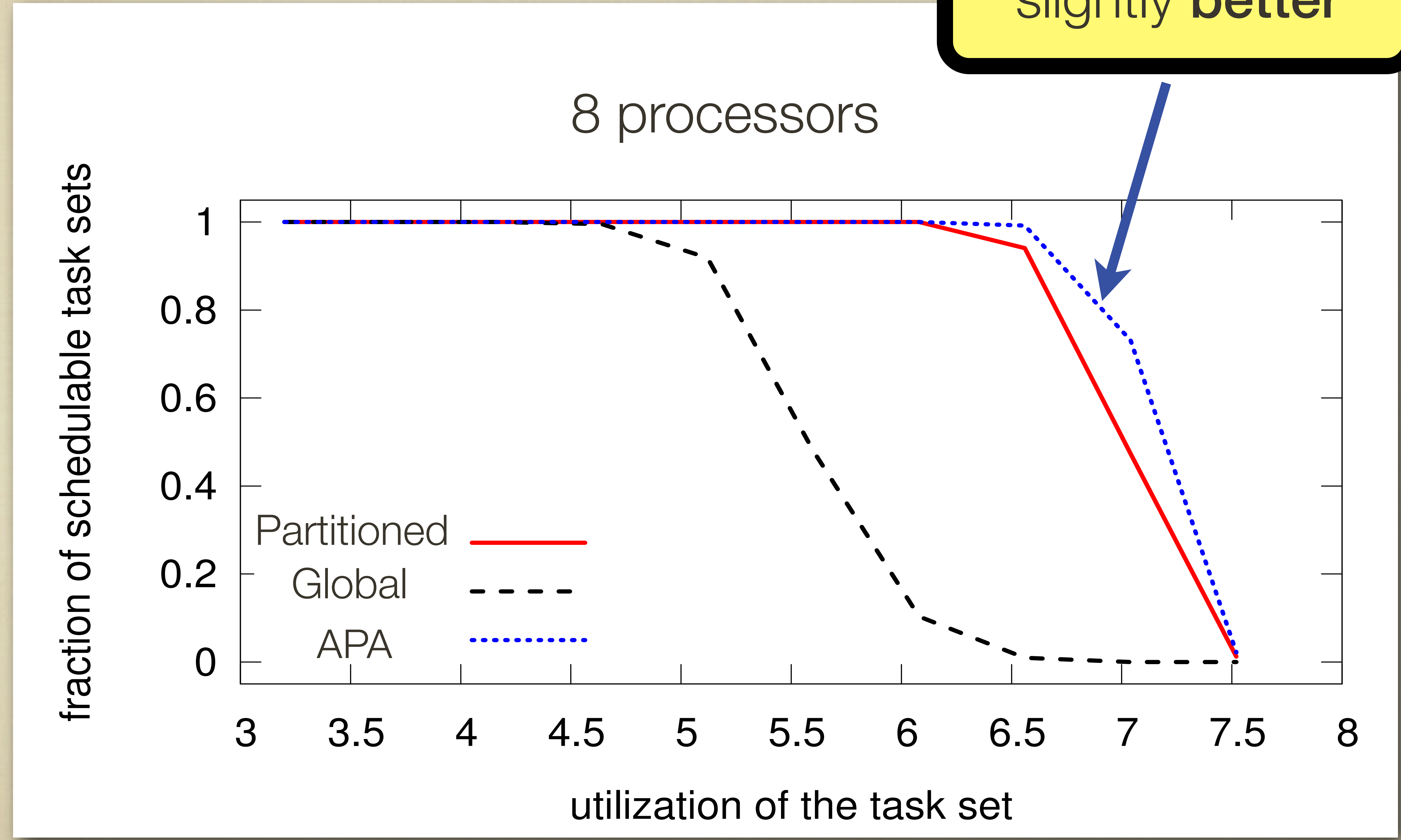
Experiment 2: partitioned vs. global vs. APA scheduling

Experiment 2: partitioned vs. global vs. APA scheduling



Experiment 2: partitioned vs. global vs. APA scheduling

APA performs slightly **better**



Are bigger gains possible?

Are bigger gains possible?

- Workloads that benefit from APA scheduling

Low-utilization tasks with **constrained** deadlines

+

High-utilization tasks with **implicit** deadlines

Are bigger gains possible?

- Workloads that benefit from APA scheduling

Low-utilization tasks with **constrained** deadlines	+	**High-utilization** tasks with **implicit** deadlines

- Under Linux scheduler design
    - Higher-priority tasks never make room for lower-priority tasks
    - Can we have **better migration rules**?

Open questions

- APA **feasibility analysis**
- **Optimal APA assignment** versus (or with) optimal priority assignment
- **Dynamic APAs** (APAs vary over time)
  - Generalize semi-partitioning as well

Summary

APA scheduling **strictly dominates** global, clustered, and partitioned JLFP scheduling.

We can derive schedulability guarantees for APA schedulers by **reduction** to global sub-problems (can reuse **any global analysis**).

APA scheduling helps **improve** schedulability.
We can do much better, many of **open questions**.

Thank you. Questions?