# An Optimal Semi-Partitioned Scheduler Assuming Arbitrary Affinity Masks

Sergey Voronov, James H. Anderson

The University of North Carolina at Chapel Hill
Department of Computer Science

39th IEEE Real-Time Systems Symposium
Nashville, Tennessee, USA

December 14, 2018

# Introduction

## Motivation

Peter Zijlstra's keynote talk (ECRTS'17).

# Introduction

## Motivation

Peter Zijlstra's keynote talk (ECRTS'17).

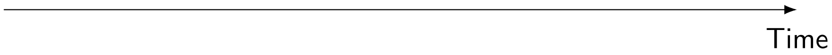
## Problem (Informal)

How to schedule a task set on an identical multiprocessor if some tasks have affinity constraints, so they can be scheduled only on specific cores?

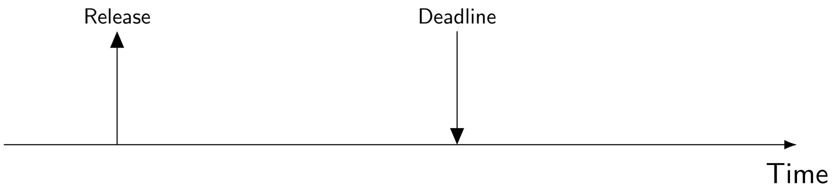
Outline:

- Necessary definitions
- Feasibility test
- Affinity reduction
- AM-Red scheduler

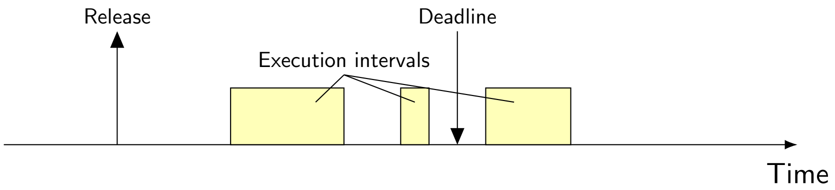
# Soft Real-Time



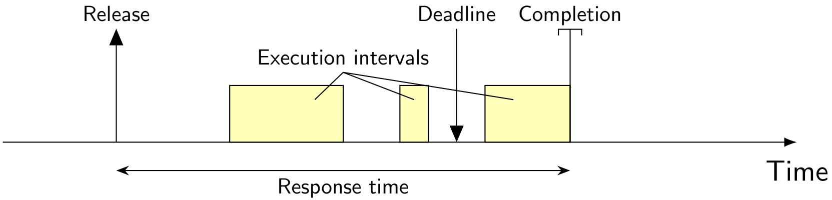
# Soft Real-Time



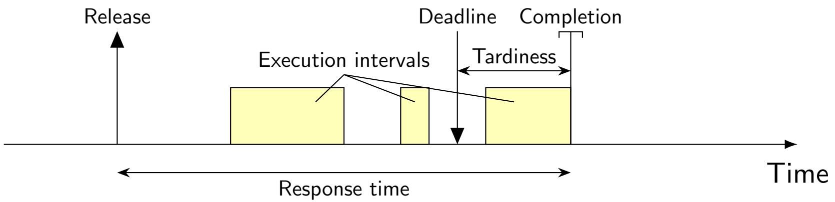
# Soft Real-Time



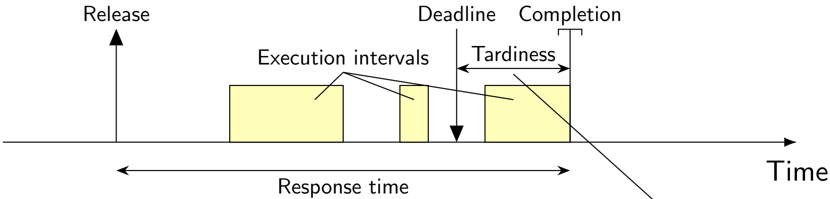
# Soft Real-Time



# Soft Real-Time



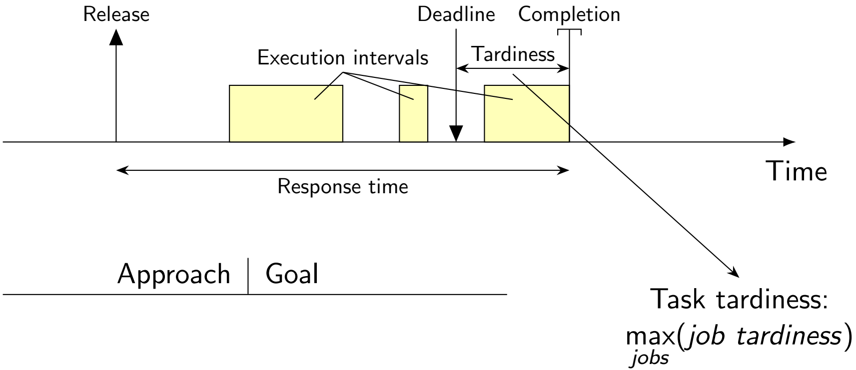
# Soft Real-Time




Task tardiness:

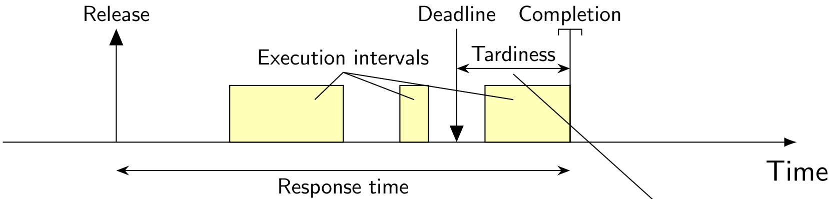
$$\max_{jobs}(\textit{job tardiness})$$

# Soft Real-Time




| Approach | Goal |
|---|---|

# Soft Real-Time

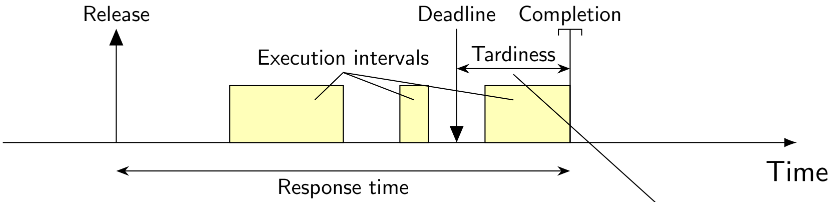



| Approach | Goal |
|---|---|
| Hard Real-Time | no tardiness at all |

Task tardiness:

$$\max_{jobs}(\textit{job tardiness})$$

# Soft Real-Time




| Approach | Goal |
|---|---|
| Hard Real-Time | no tardiness at all |
| Soft Real-Time | bounded tardiness for every task |

Task tardiness:

$$\max_{jobs}(\textit{job tardiness})$$

# Affinity Masks

## Affinity Mask

A set of cores where a task can be scheduled.

# Affinity Masks

## Affinity Mask

A set of cores where a task can be scheduled.

Usage:

- performance improvement (ensure shared cache)
- load balancing
- task isolation

# Affinity Masks

## Affinity Mask

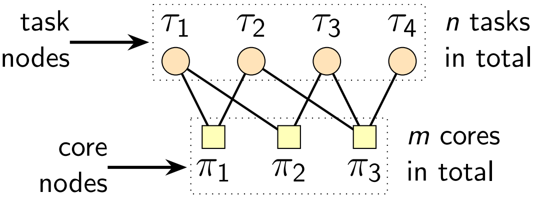
A set of cores where a task can be scheduled.

Usage:

- performance improvement (ensure shared cache)
- load balancing
- task isolation

## Affinity Graph (AG)

Bipartite graph that shows affinity masks.



# Feasibility Test

## Task Set Feasibility Problem (Formal)

Can any algorithm schedule a given task set $\tau$ of size $n$ with a given affinity masks set on an identical multiprocessor with $m$ cores?

# Feasibility Test

## Task Set Feasibility Problem (Formal)

Can any algorithm schedule a given task set $\tau$ of size $n$ with a given affinity masks set on an identical multiprocessor with $m$ cores?

## Feasibility without affinities

$\sum$ utilizations of tasks $\leq m$

# Feasibility Test

## Task Set Feasibility Problem (Formal)

Can any algorithm schedule a given task set $\tau$ of size $n$ with a given affinity masks set on an identical multiprocessor with $m$ cores?

## Feasibility without affinities

$\sum$ utilizations of tasks $\leq m$

## Feasibility with affinities

For every subset of tasks $\tau' \subset \tau$:

$\sum_{\tau'}$ utilizations of tasks $\leq$ size of $\tau'$ affinities union

# Feasibility Test

## Task Set Feasibility Problem (Formal)

Can any algorithm schedule a given task set $\tau$ of size $n$ with a given affinity masks set on an identical multiprocessor with $m$ cores?

## Feasibility without affinities

$\sum$ utilizations of tasks $\leq m$

## Existing test[#]

LP-based, $\tilde{O}(mn(m+n)^{2.9})$

## Feasibility with affinities

For every subset of tasks $\tau' \subset \tau$:

$\sum_{\tau'}$ utilizations of tasks $\leq$ size of $\tau'$ affinities union

## Proposed test

Flow-based, $\tilde{O}(mn\sqrt{m+n})$

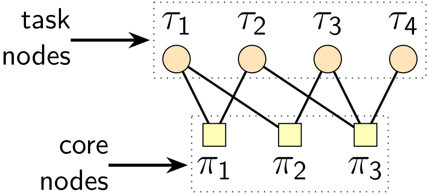
[#] S. Baruah and B. Brandenburg, "Multiprocessor feasibility analysis of recurrent task systems with specified processor affinities", RTSS'13.

# Feasibility Test: Overview

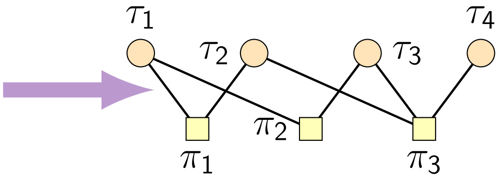
Proposed test: Max Flow is $\sum_{\tau_i \in \tau} U_i \Rightarrow$ task set is feasible.

# Feasibility Test: Overview

Proposed test: Max Flow is $\sum_{\tau_i \in \tau} U_i \Rightarrow$ task set is feasible.



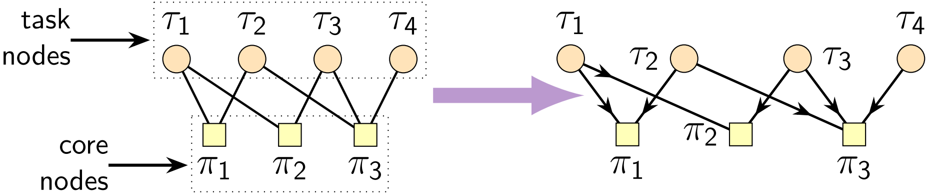

Affinity graph (AG).




AG-based flow network.

# Feasibility Test: Overview

Proposed test: Max Flow is $\sum_{\tau_i \in \tau} U_i \Rightarrow$ task set is feasible.

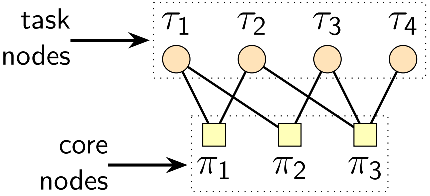



Affinity graph (AG).

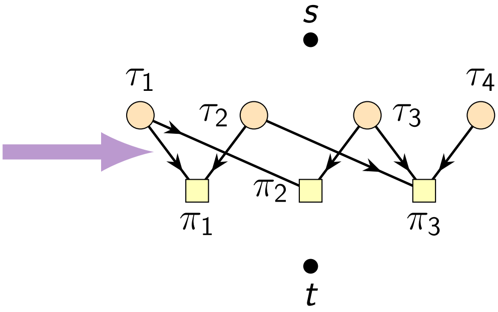
AG-based flow network.

# Feasibility Test: Overview

Proposed test: Max Flow is $\sum_{\tau_i \in \tau} U_i \Rightarrow$ task set is feasible.



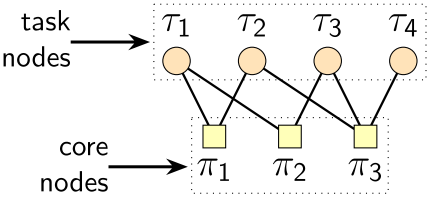

Affinity graph (AG).



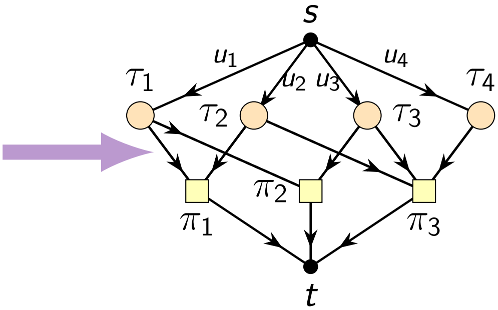

AG-based flow network.

# Feasibility Test: Overview

Proposed test: Max Flow is $\sum_{\tau_i \in \tau} U_i \Rightarrow$ task set is feasible.



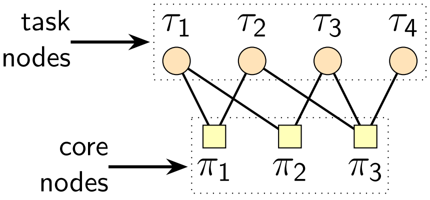

Affinity graph (AG).



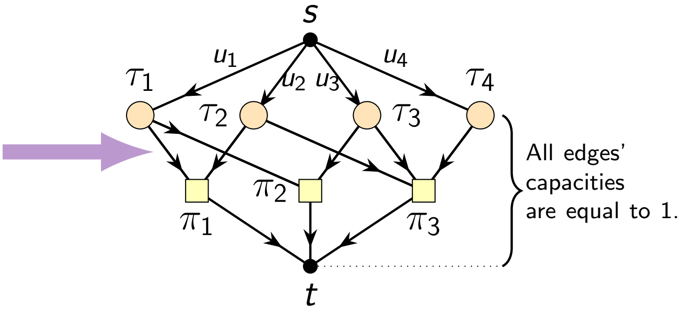

AG-based flow network.

# Feasibility Test: Overview

Proposed test: Max Flow is $\sum_{\tau_i \in \tau} U_i \Rightarrow$ task set is feasible.




Affinity graph (AG).

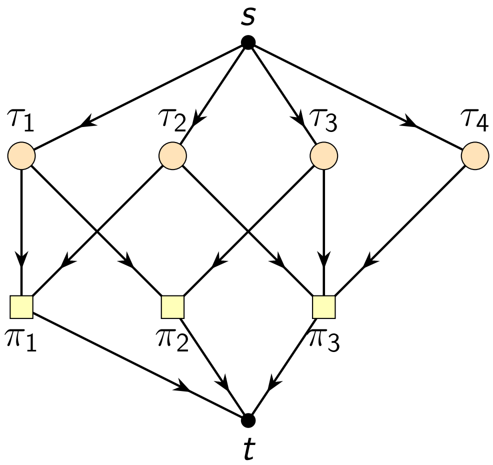



AG-based flow network.

# Affinity Reduction

**General idea**

Edge without flow can be removed from AG.



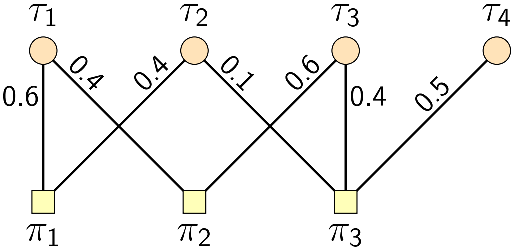
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



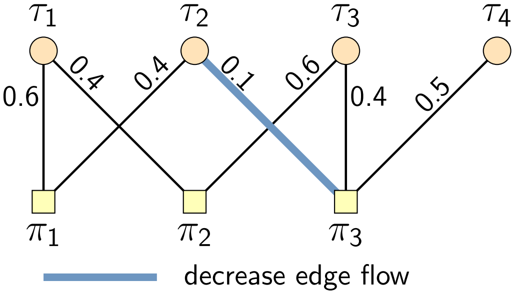
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



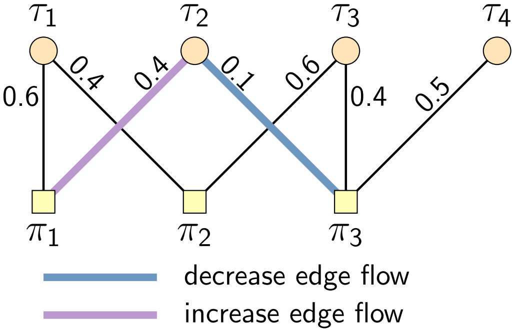
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



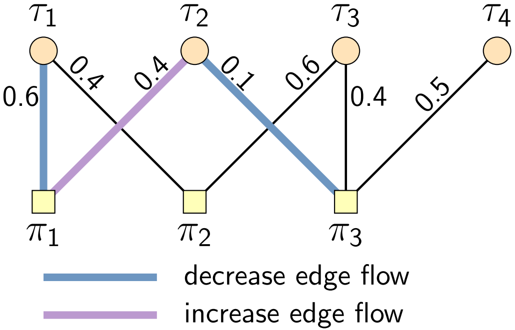
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



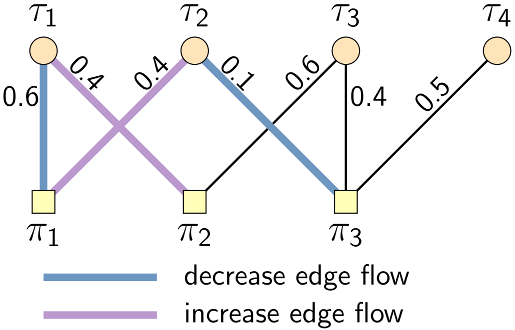
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



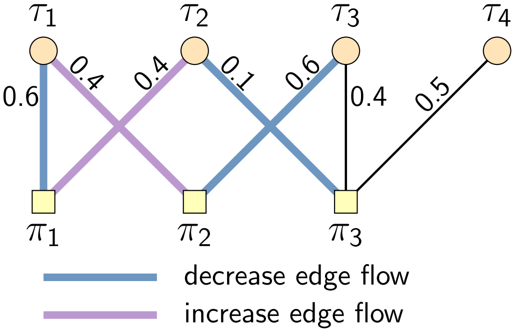
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



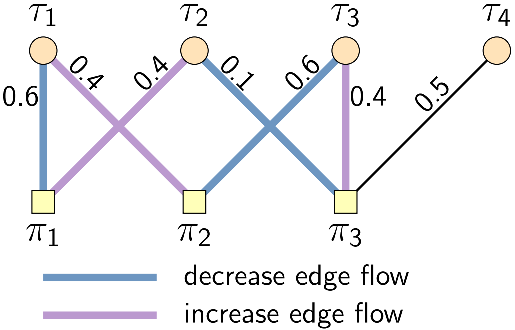
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



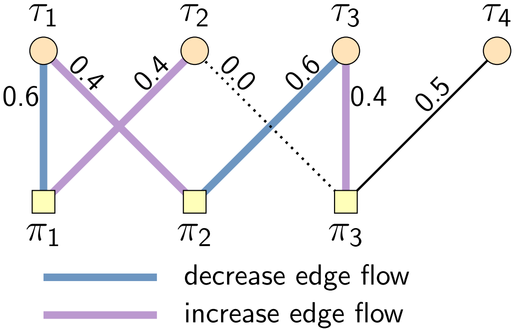
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



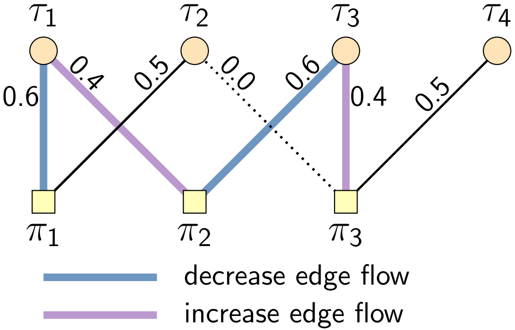
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



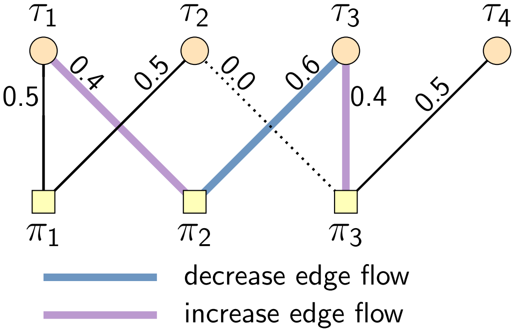
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



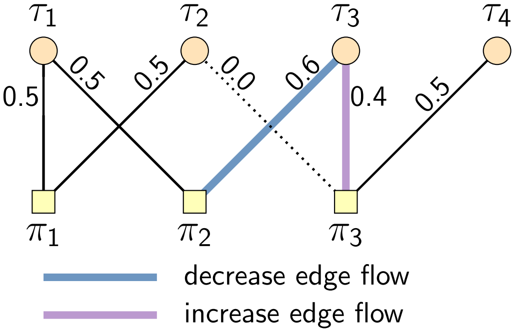
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



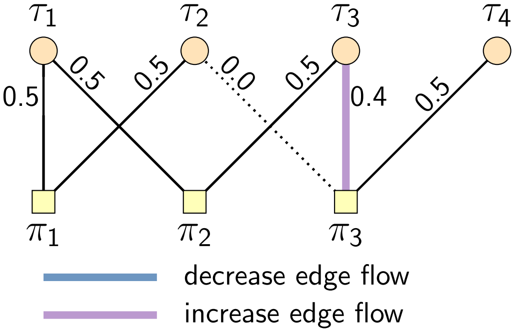
# Affinity Reduction

## General idea

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



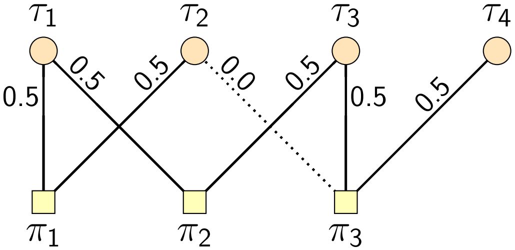
# Affinity Reduction

**General idea**

Edge without flow can be removed from AG.

Change rules:

- Total flow from a task should not change
- Total flow to a core should not change



# AM-Red (Affinity Mask Reduction): Overview

- Compute Max Flow (checks feasibility)

# AM-Red (Affinity Mask Reduction): Overview

- Compute Max Flow (checks feasibility)
- Apply affinity reduction (ensures at most $m-1$ migrating tasks)

# AM-Red (Affinity Mask Reduction): Overview

tasks which are scheduled on at least two different cores

- Compute Max Flow (checks feasibility)
- Apply affinity reduction (ensures at most $m - 1$ migrating tasks)

# AM-Red (Affinity Mask Reduction): Overview

tasks which are scheduled on at least two different cores

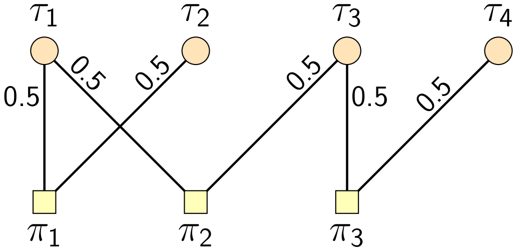
- Compute Max Flow (checks feasibility)
- Apply affinity reduction (ensures at most $m - 1$ migrating tasks)
- Construct a schedule template of size $|F|$ (ensures a tardiness bound of $|F|$)

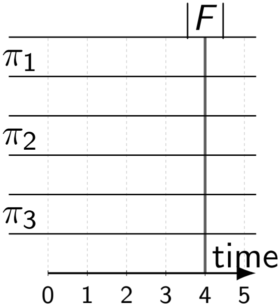
# AM-Red (Affinity Mask Reduction): Overview

tasks which are scheduled on at least two different cores

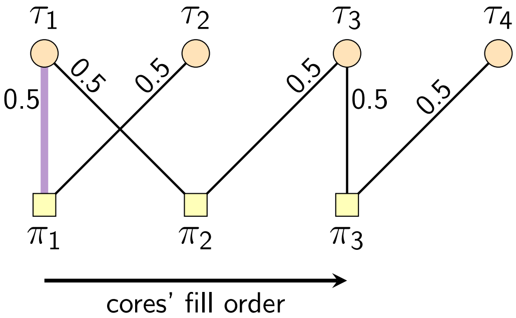
- Compute Max Flow (checks feasibility)
- Apply affinity reduction (ensures at most $m - 1$ migrating tasks)
- Construct a schedule template of size $|F|$ (ensures a tardiness bound of $|F|$)
- Repeat template to get actual schedule

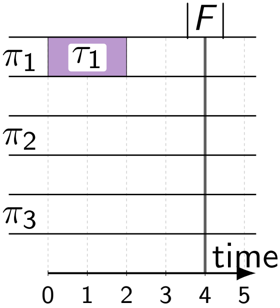
# AM-Red: Template Construction



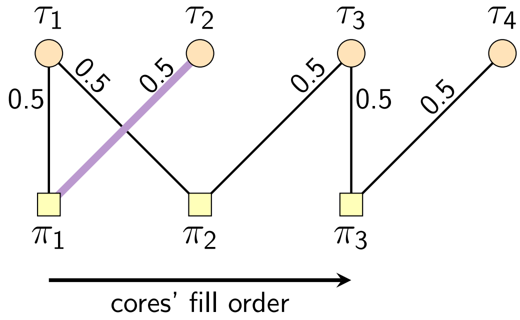



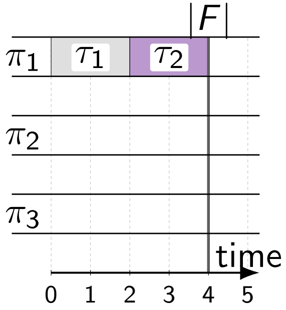
# AM-Red: Template Construction



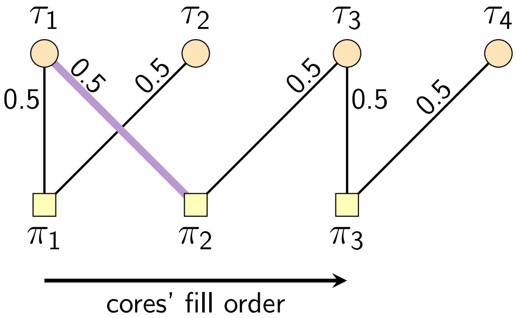



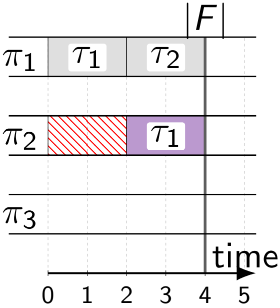
# AM-Red: Template Construction



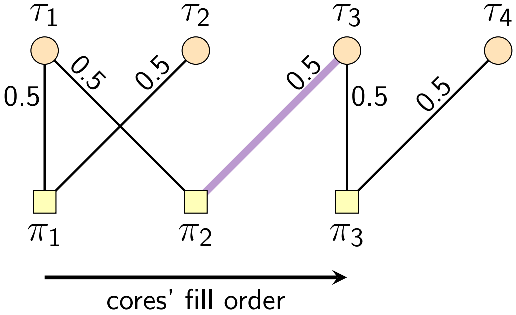



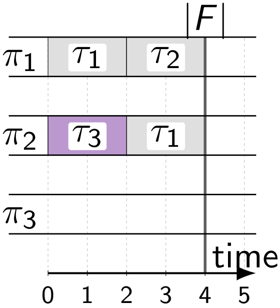
# AM-Red: Template Construction



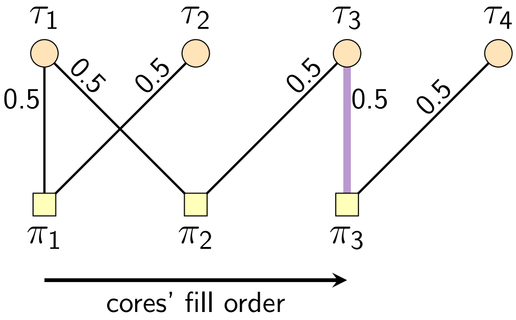



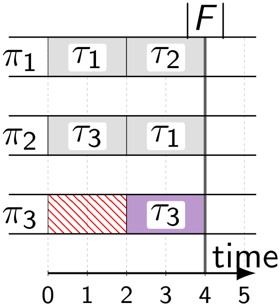
# AM-Red: Template Construction



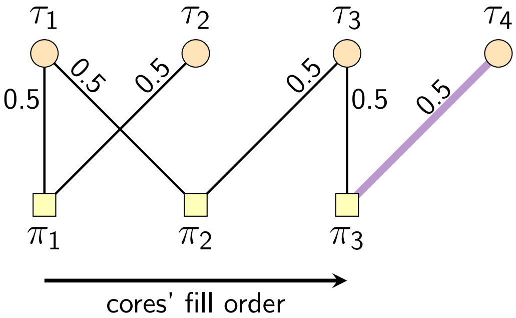



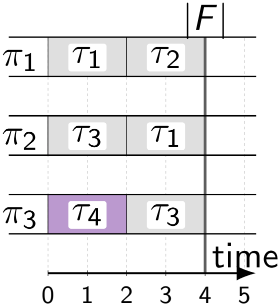
# AM-Red: Template Construction



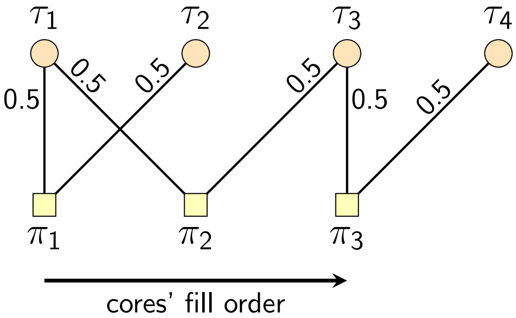



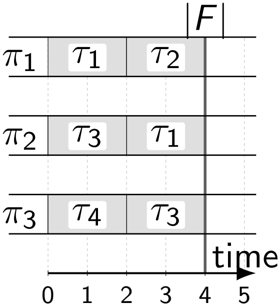
# AM-Red: Template Construction



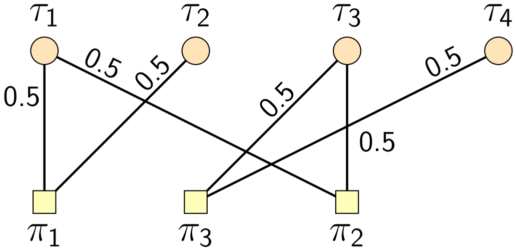



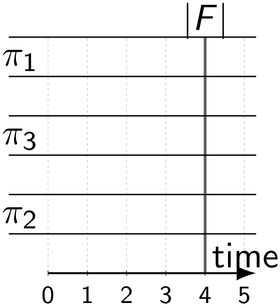
# AM-Red: Template Construction



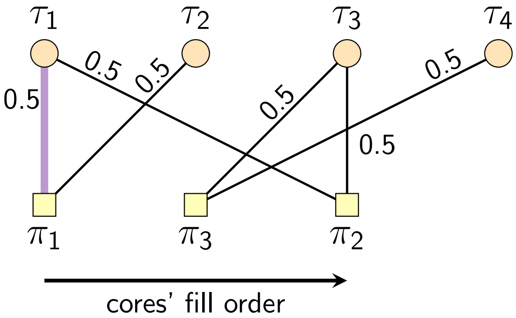



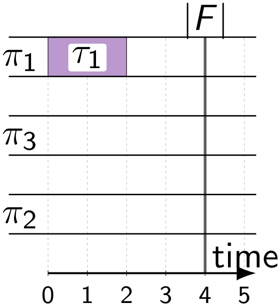
# AM-Red: Template Construction (mixed cores order)



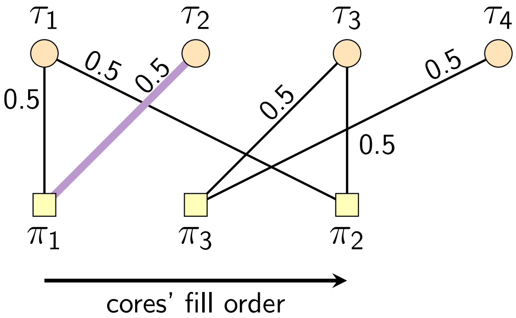



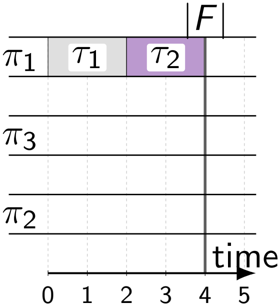
# AM-Red: Template Construction (mixed cores order)



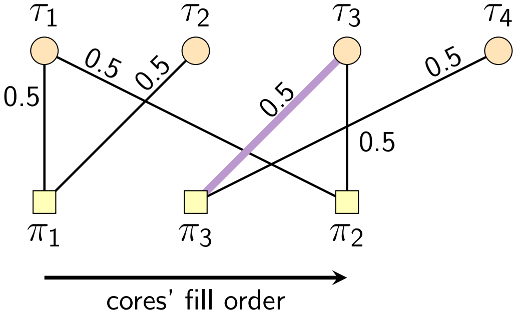



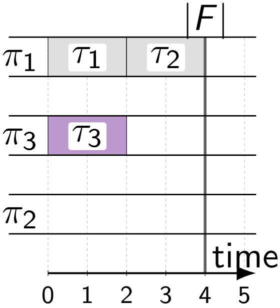
# AM-Red: Template Construction (mixed cores order)



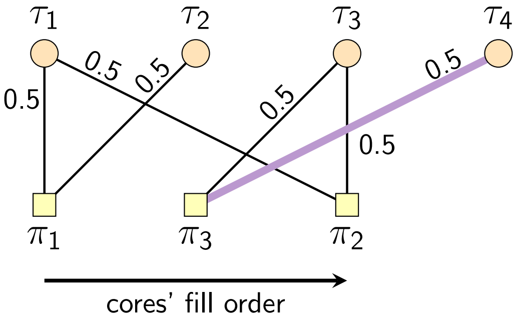



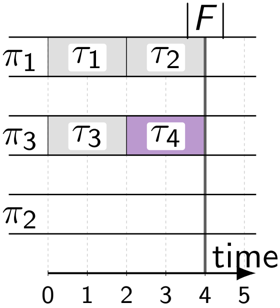
# AM-Red: Template Construction (mixed cores order)



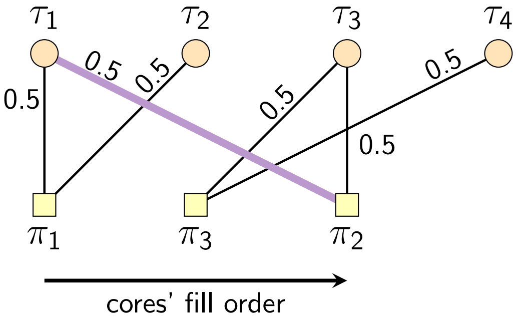



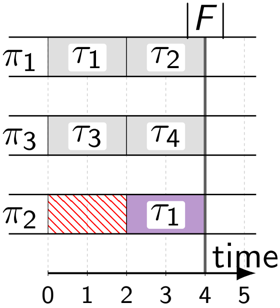
# AM-Red: Template Construction (mixed cores order)



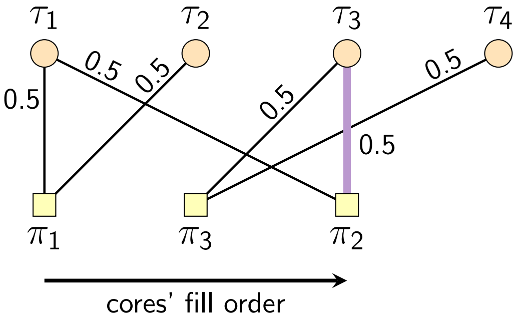



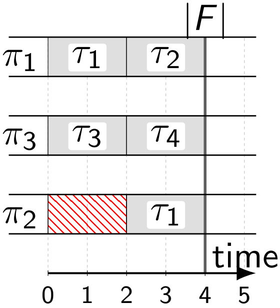
# AM-Red: Template Construction (mixed cores order)



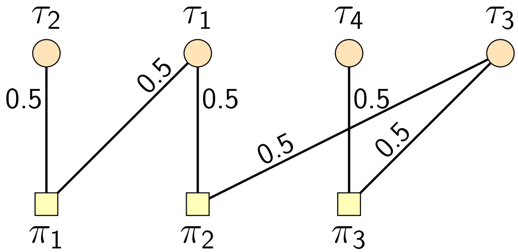



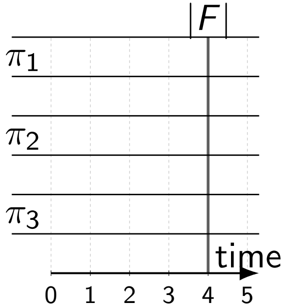
# AM-Red: Template Construction (mixed cores order)



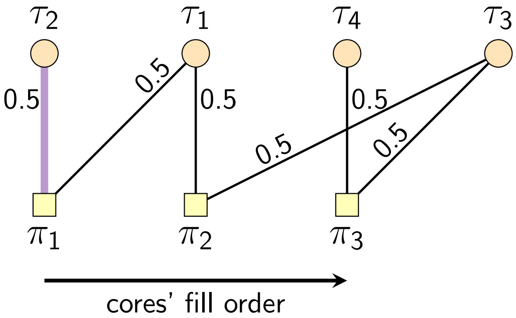



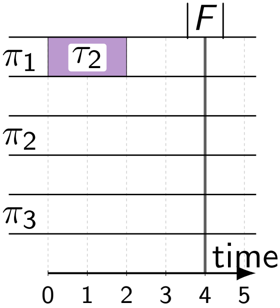
# AM-Red: Template Construction (mixed tasks order)



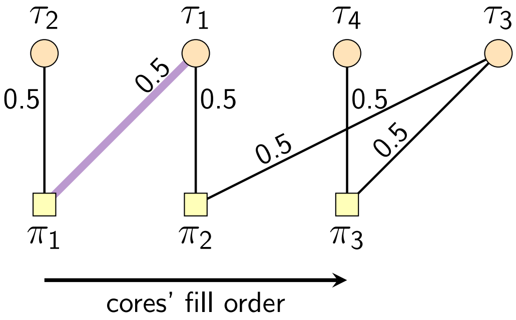



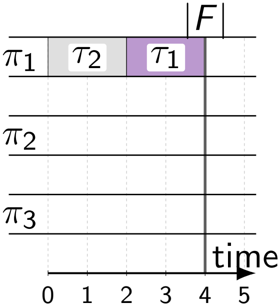
# AM-Red: Template Construction (mixed tasks order)



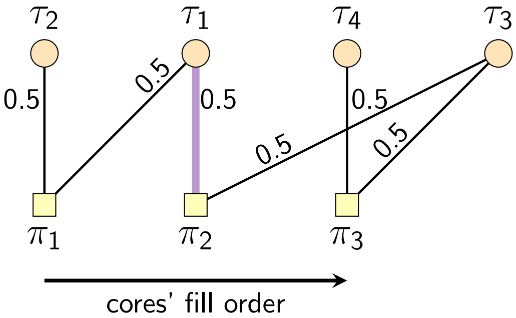



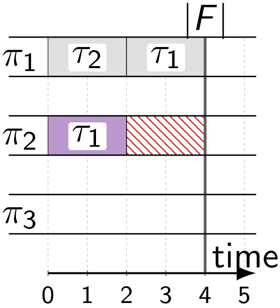
# AM-Red: Template Construction (mixed tasks order)



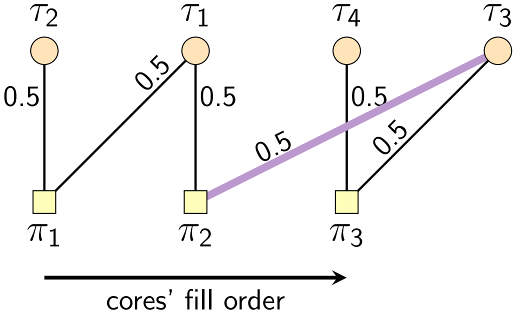



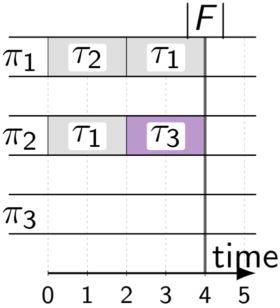
# AM-Red: Template Construction (mixed tasks order)



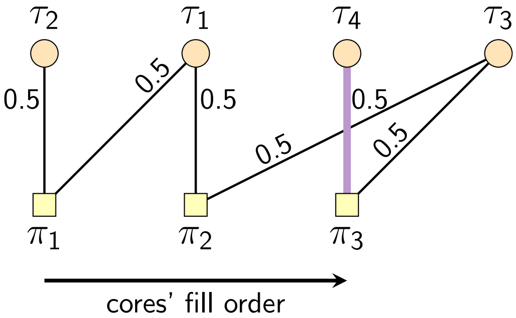



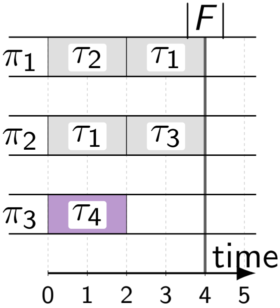
# AM-Red: Template Construction (mixed tasks order)



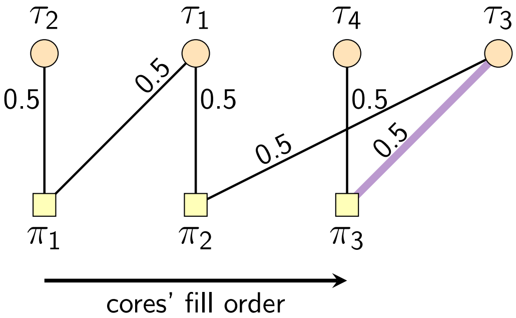



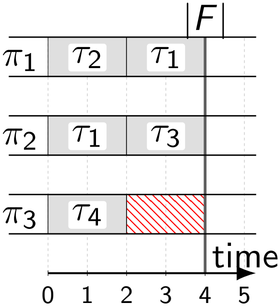
# AM-Red: Template Construction (mixed tasks order)



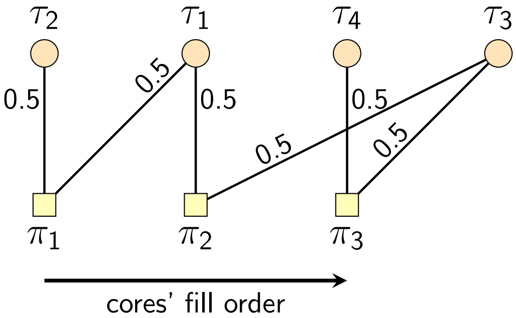



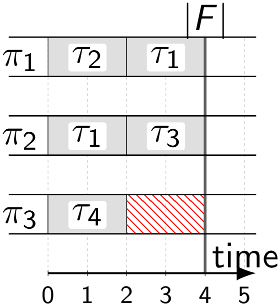
# AM-Red: Template Construction (mixed tasks order)



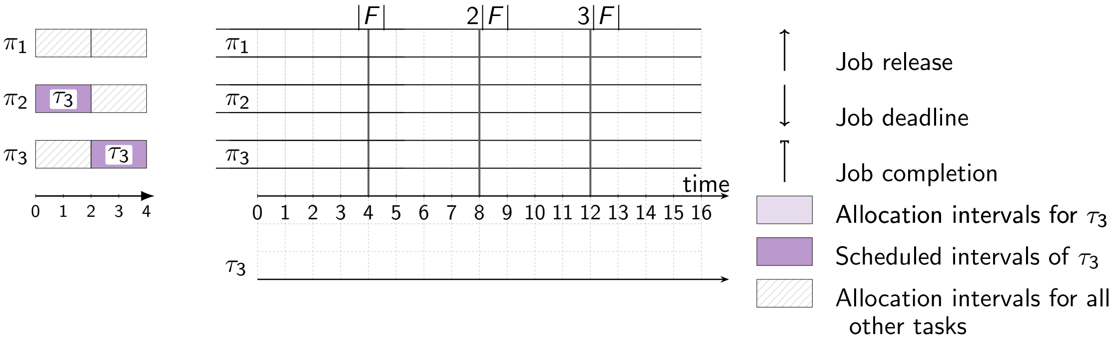



# AM-Red: Template Construction (mixed tasks order)

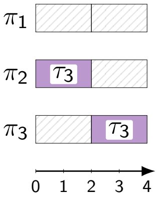


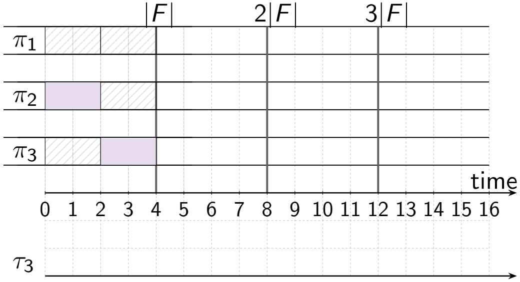



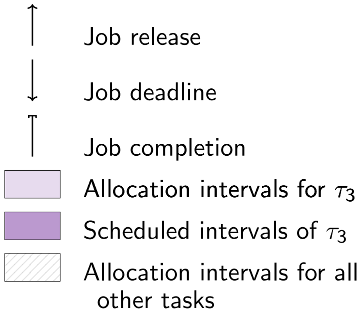
# AM-Red: Final Schedule



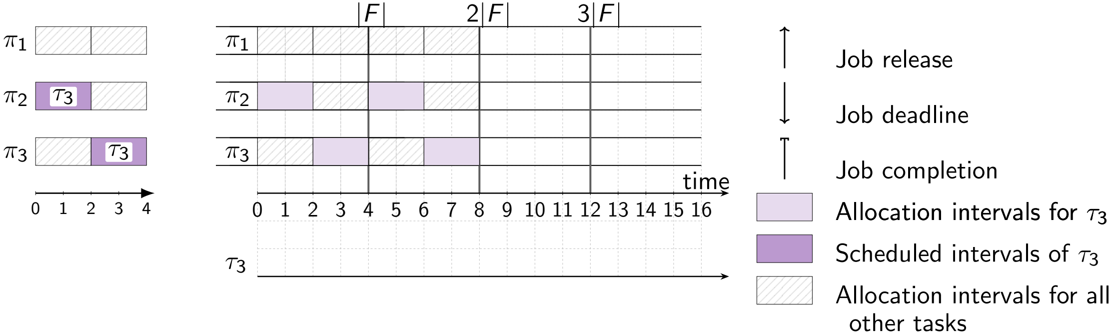
# AM-Red: Final Schedule



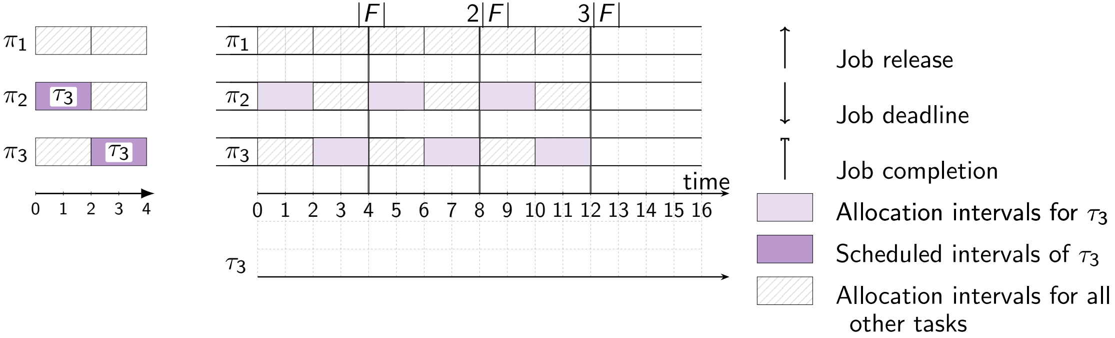



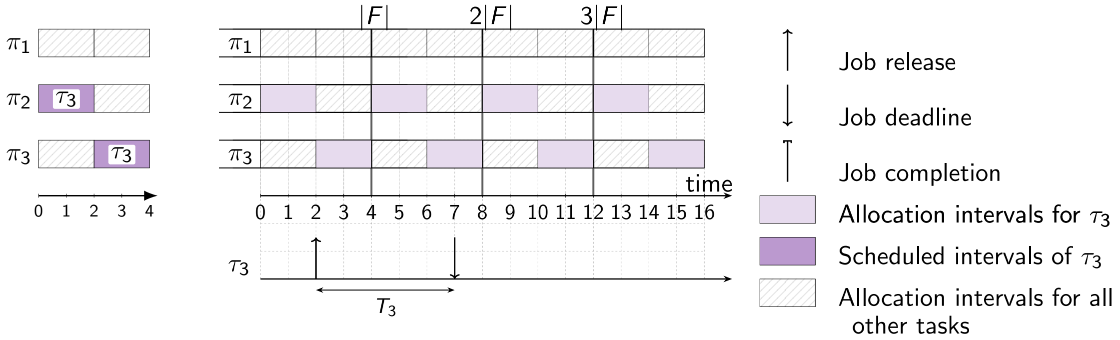
# AM-Red: Final Schedule



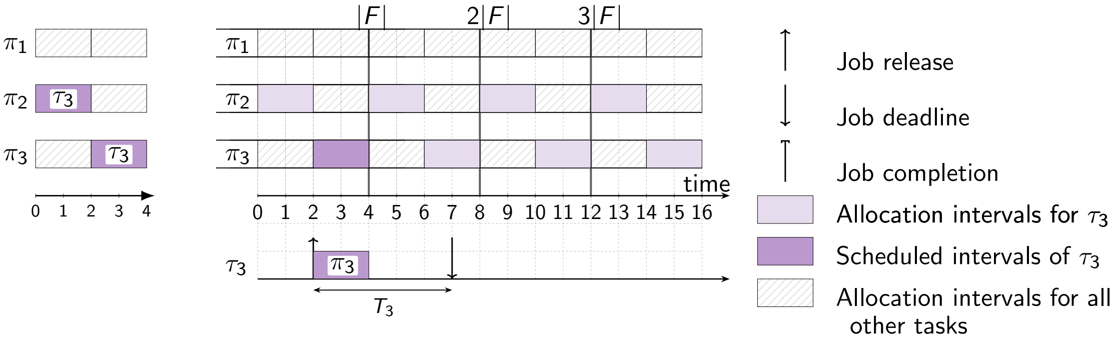
# AM-Red: Final Schedule



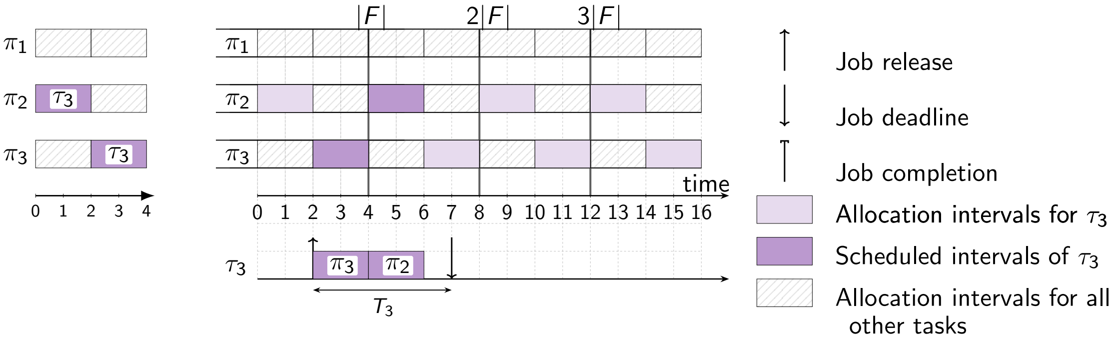
# AM-Red: Final Schedule



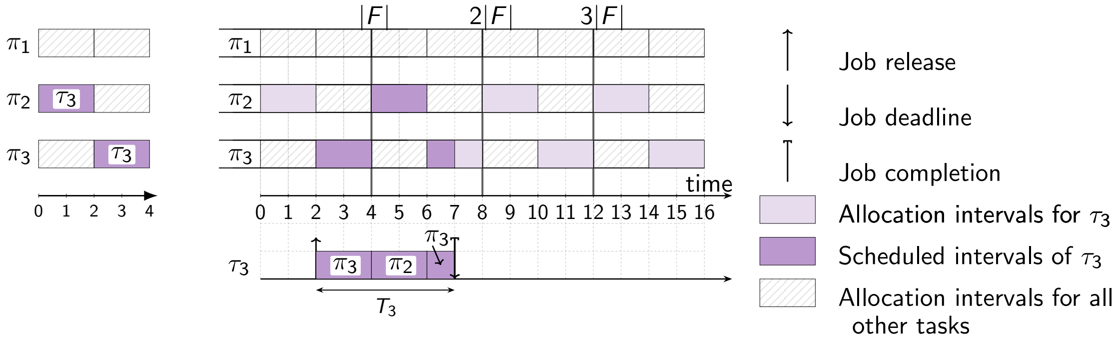
# AM-Red: Final Schedule



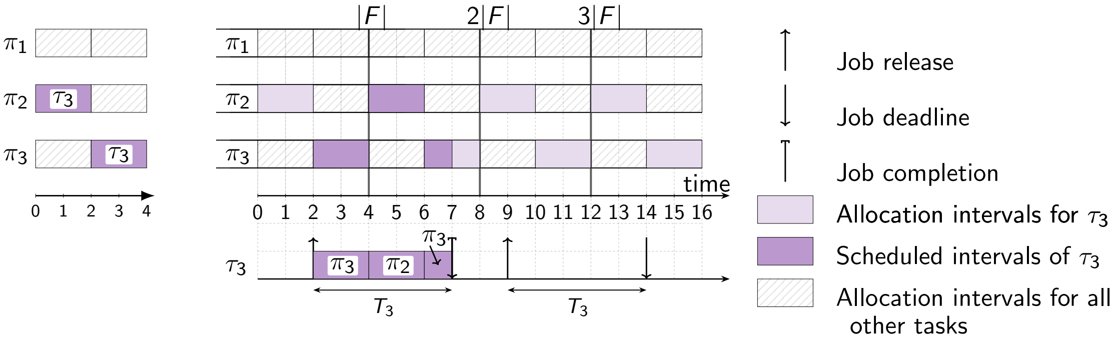
# AM-Red: Final Schedule



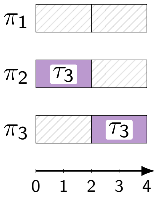
# AM-Red: Final Schedule

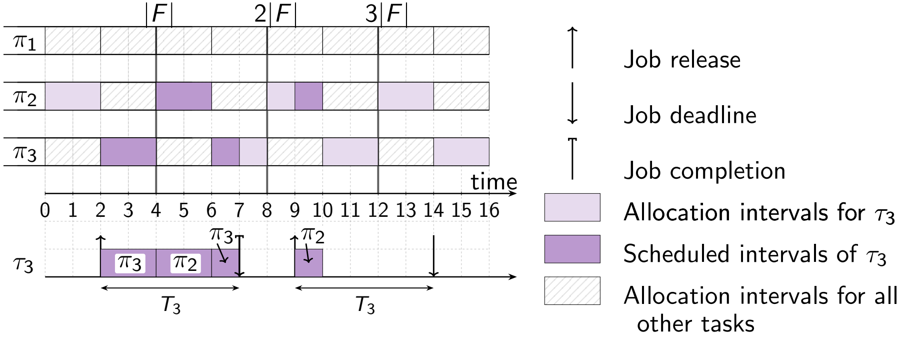


# AM-Red: Final Schedule

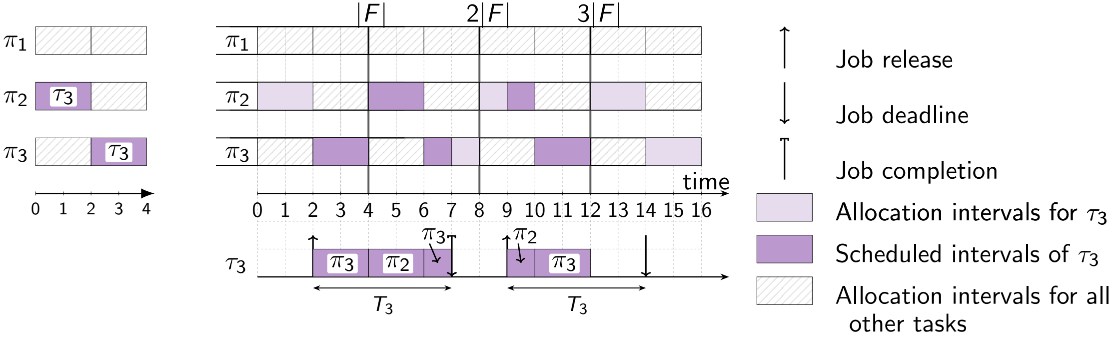


# AM-Red: Final Schedule

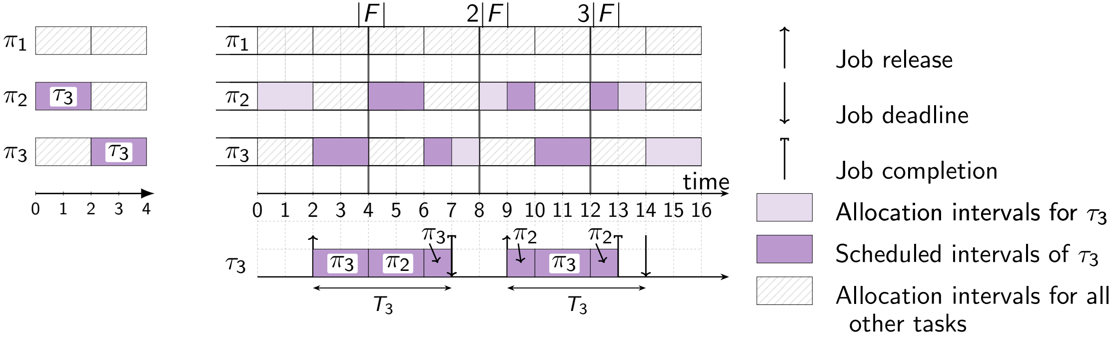


# AM-Red: Final Schedule



# AM-Red: Final Schedule



# Conclusion

- SRT-optimal ($|F|$ is a tardiness bound)

# Conclusion

- SRT-optimal ($|F|$ is a tardiness bound)
- At most $2m - 2$ migrations per template

# Conclusion

- SRT-optimal ($|F|$ is a tardiness bound)
- At most $2m - 2$ migrations per template
- Total complexity is $O(m^2 n^2)$

# Conclusion

- SRT-optimal ($|F|$ is a tardiness bound)
- At most $2m - 2$ migrations per template
- Total complexity is $O(m^2 n^2)$
  ($O(m + n)$ for hierarchical masks)

# Conclusion

- SRT-optimal ($|F|$ is a tardiness bound)
- At most $2m-2$ migrations per template
- Total complexity is $O(m^2 n^2)$
  ($O(m+n)$ for hierarchical masks)

Results:

- Proposed exact feasibility test.
- Developed an affinity reduction function.
- AM-Red is an SRT-optimal semi-partitioned scheduler.