The impact of heterogeneity on master-slave scheduling

Jean-François PINEAU, Yves ROBERT and Frédéric VIVIEN

Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon, France

Jean-Francois.Pineau@ens-lyon.fr

http://graal.ens-lyon.fr/~jfpineau

ALPAGE June 13, 2006

Outline

Outline

Background on Scheduling

The tasks

described by:

- their amount of computation
- their amount of communication
- their release date r_i

Notation for their date of end of execution: C_i

The processors

Background on Scheduling

The tasks

described by:

- identical amount of computation
- identical amount of communication
- their release date r_i

Notation for their date of end of execution: C_i

The processors

Background on Scheduling

The tasks

The processors

described by:

- their computation speed w_j
- the speed of their communication links c_j

Background on Scheduling

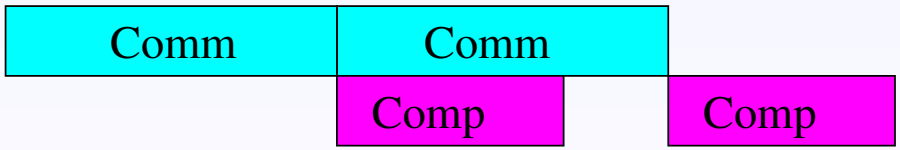
The tasks

The processors

described by:

- their computation speed w_j
- the speed of their communication links c_j

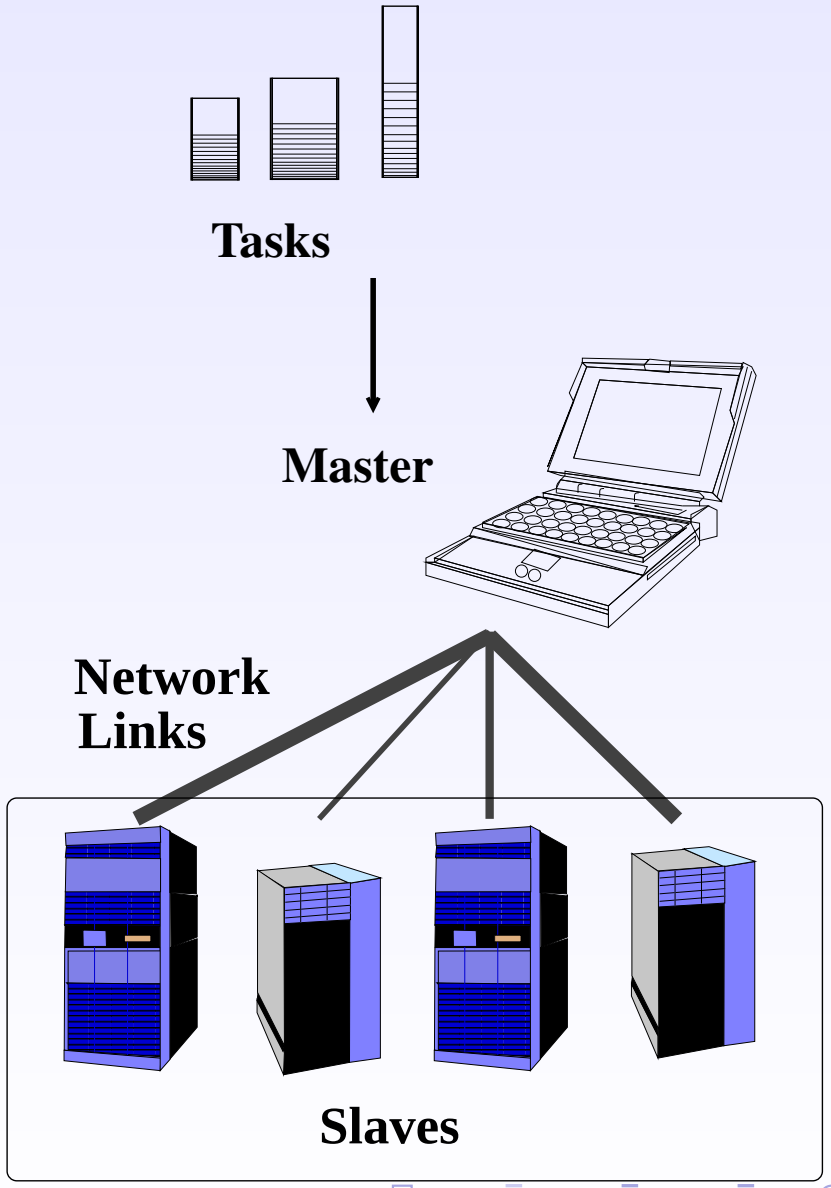
If $c_{j_0} = min\ c_j$ and $c_{j_0} > w_{j_0}$, then the optimal algorithm is trivial.



Background on Scheduling

The master

- Receive the tasks
- Send them to the processors



Background on Scheduling

Goal

Scheduling tasks onto processors

- according to the constraints,
  - of the processors
  - of the tasks
- and optimizing some objective functions
  - makespan: max C_i
  - maximum flow time: max $(C_i - r_i)$
  - average flow time: $\sum(C_i - r_i)$

Background on Scheduling

On-line scheduling

Scheduler does not know neither the total number of tasks nor their released dates

Competitive ratio

An algorithm $\mathcal{X}$ has a lower bound ρ on its competitive ratio for some objective function if there exists one problem instance such that:

$$\left(\max C_i\right)_{\mathcal{X}} \geq \rho\left(\max C_i\right)_{Opt}$$

Outline

Outline

On homogeneous platforms

Round-Robin

is an optimal algorithm to minimize **all three**

- *makespan*,
- max flow time,
- sum flow time,

for an on-line problem with release dates.

Outline

On heterogeneous platforms

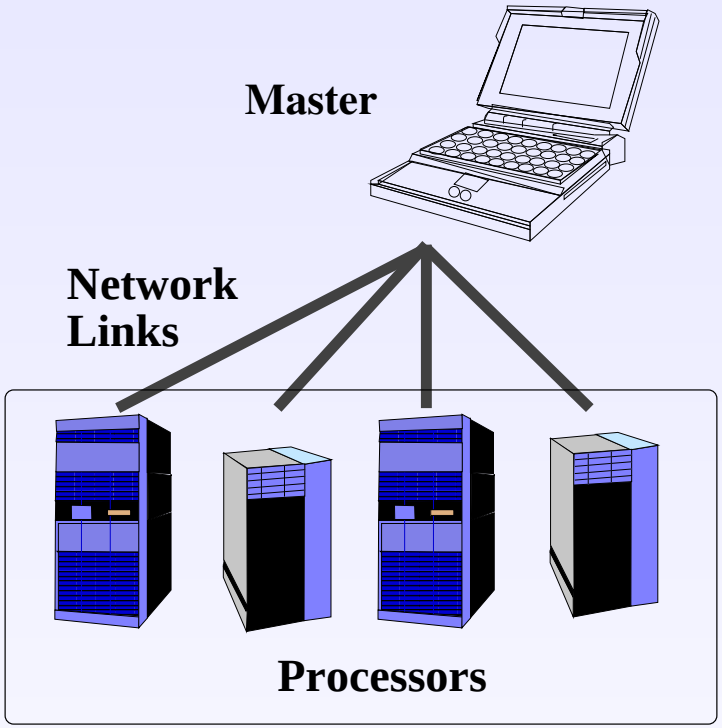
Optimal algorithm

does not exist, to minimize **one** objective function among

- *makespan*,
- max flow time,
- sum flow time,

This can be proved by an adversary method.

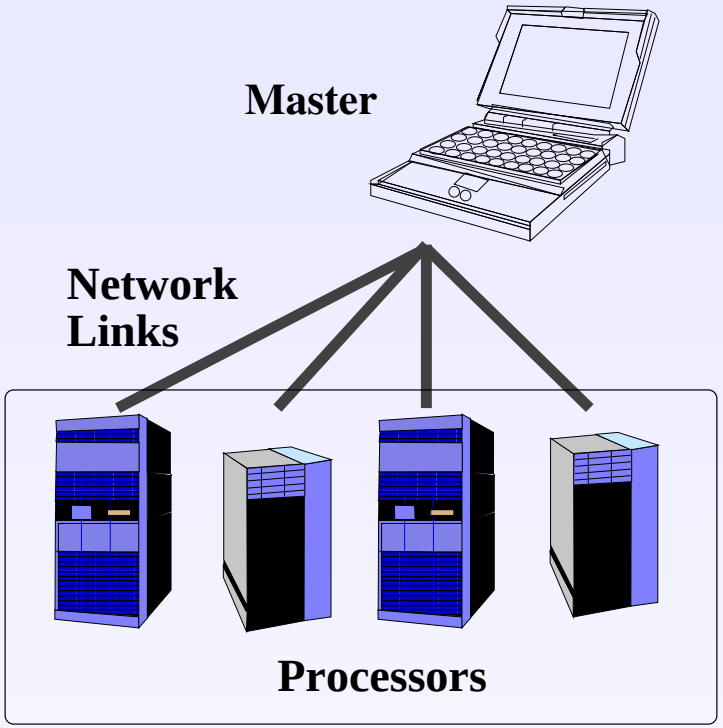
Example



Theorem

There is no scheduling algorithm for the problem $Q, MS \mid \text{online}, r_j, w_j, c_j = c \mid \max C_i$ whose competitive ratio ρ is strictly lower than $\frac{5}{4}$.

Example



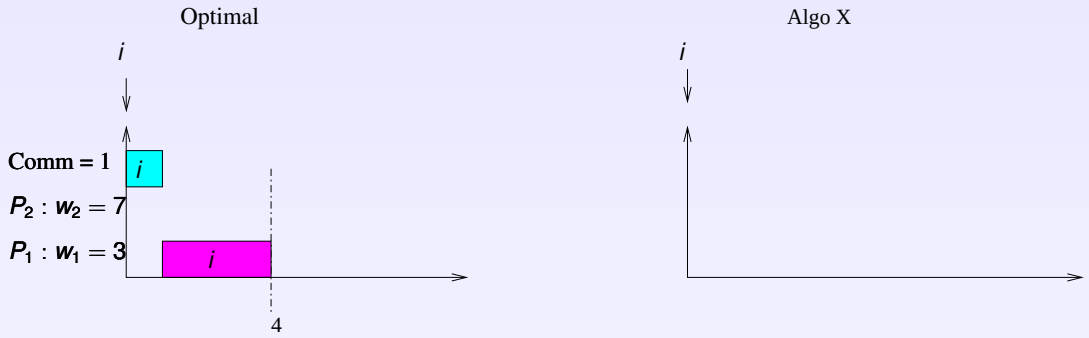
Theorem

There is no scheduling algorithm for the problem $Q, MS \mid \text{online}, r_i, w_j, c_j = c \mid \max C_i$ whose competitive ratio ρ is strictly lower than $\frac{5}{4}$.

Proof

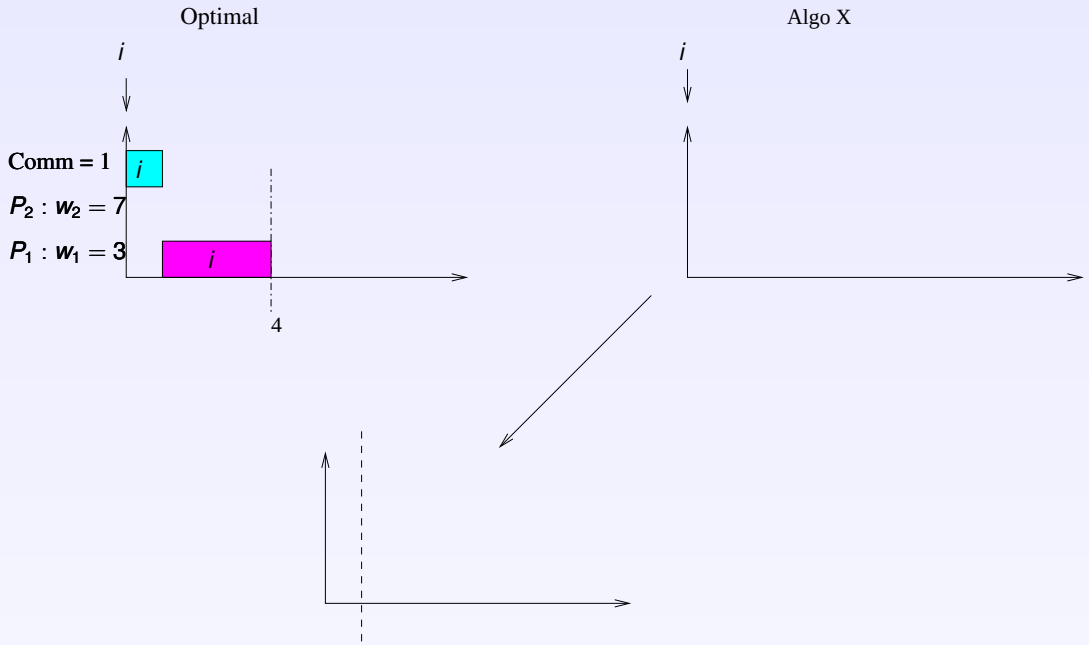
1. Suppose the existence of an on-line algorithm $\mathcal{X}$ with a competitive ratio $\rho = \frac{5}{4} - \epsilon$, with $\epsilon > 0$.
2. Let's study the behavior of $\mathcal{X}$ opposed to our adversary on a platform composed of two processors, where $w_1 = 3$, $w_2 = 7$, and $c = 1$.

Proof



Adversary sends a single task i at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

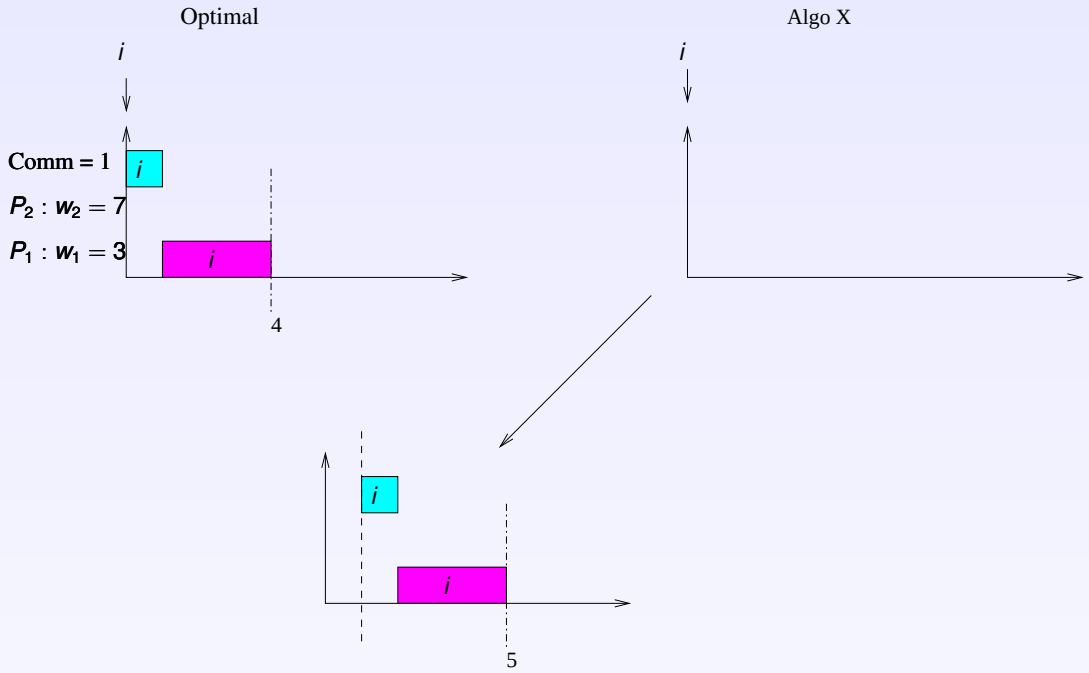
Proof



Adversary sends a single task i at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

- **competitive ratio :**

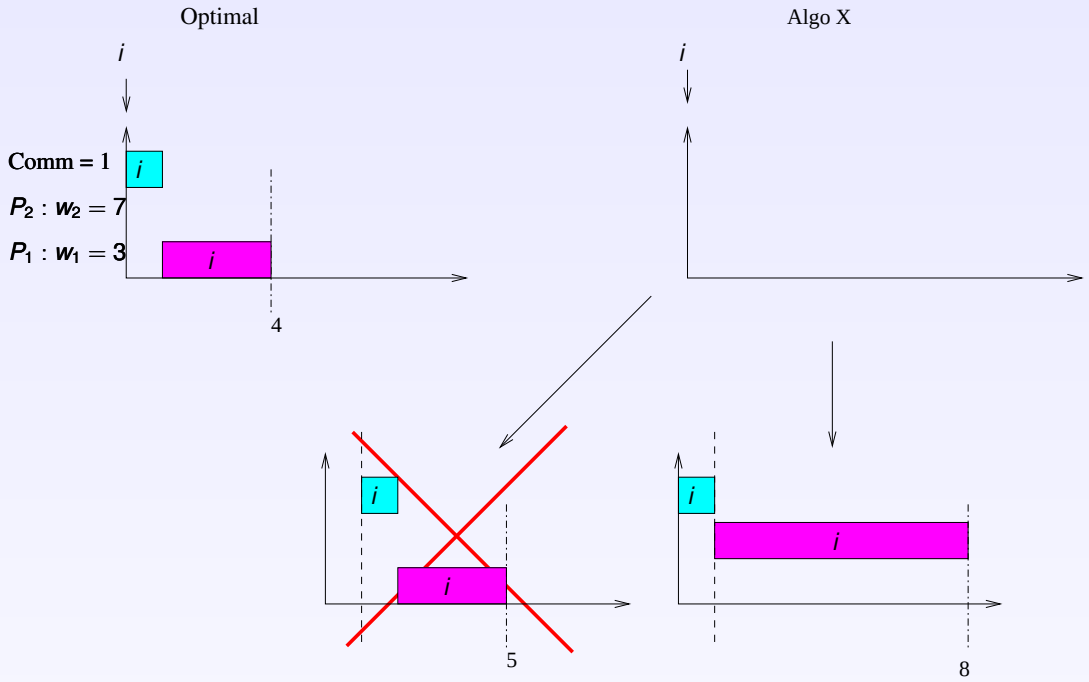
Proof



Adversary sends a single task i at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

- **competitive ratio :** $\frac{t_1+c+w_1}{4} = \frac{5}{4} > \rho$

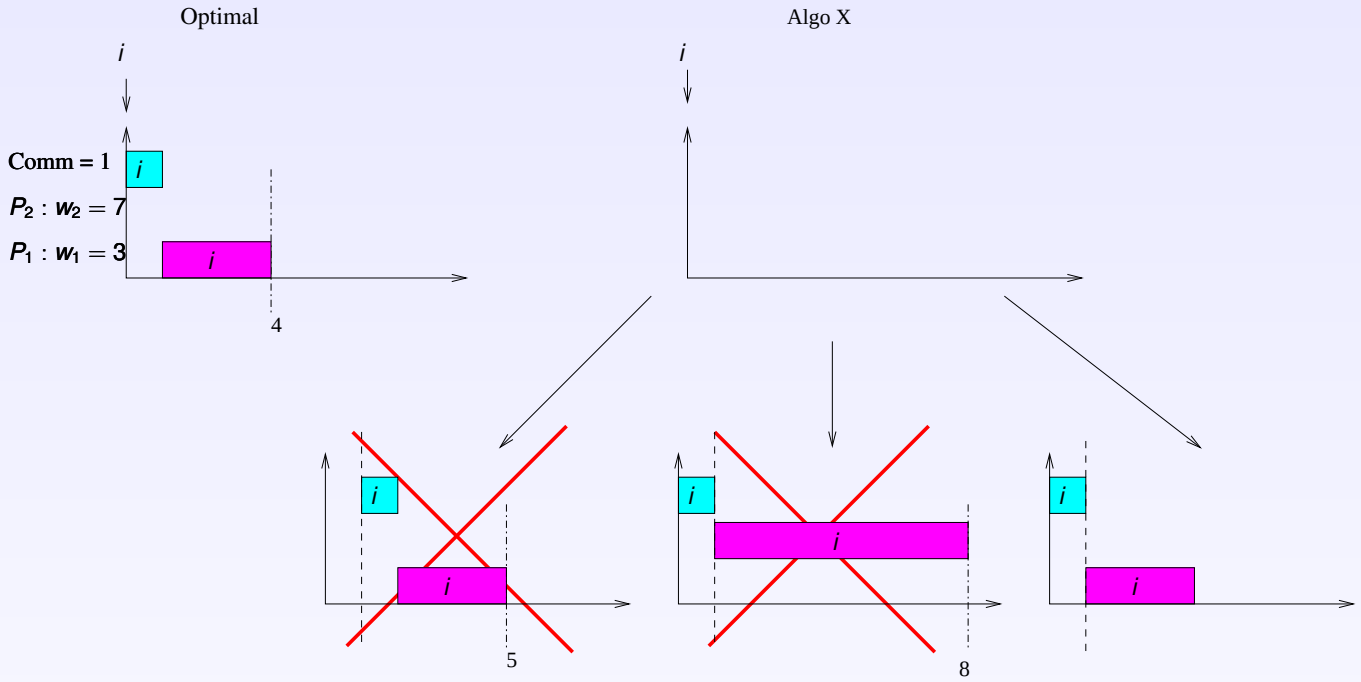
Proof



Adversary sends a single task i at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

- **competitive ratio :** $\frac{c+w_2}{4} = 2 > \rho$

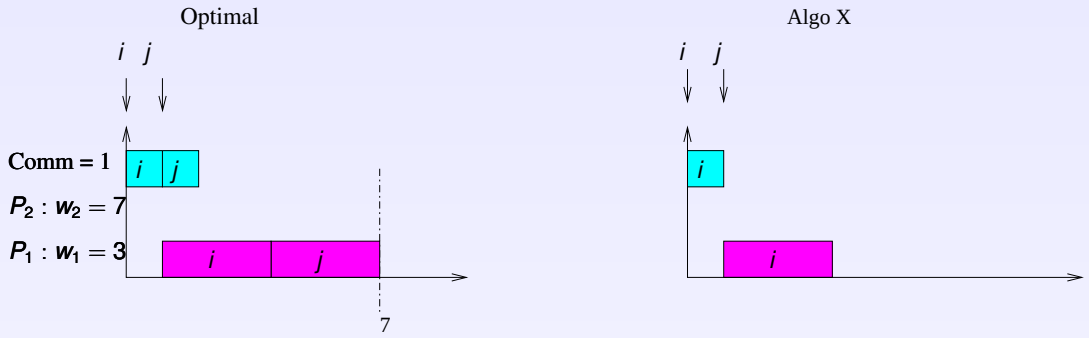
Proof



Adversary sends a single task i at time 0: best makespan = 4
At time $t_1 = c$, we check the decision of $\mathcal{X}$.

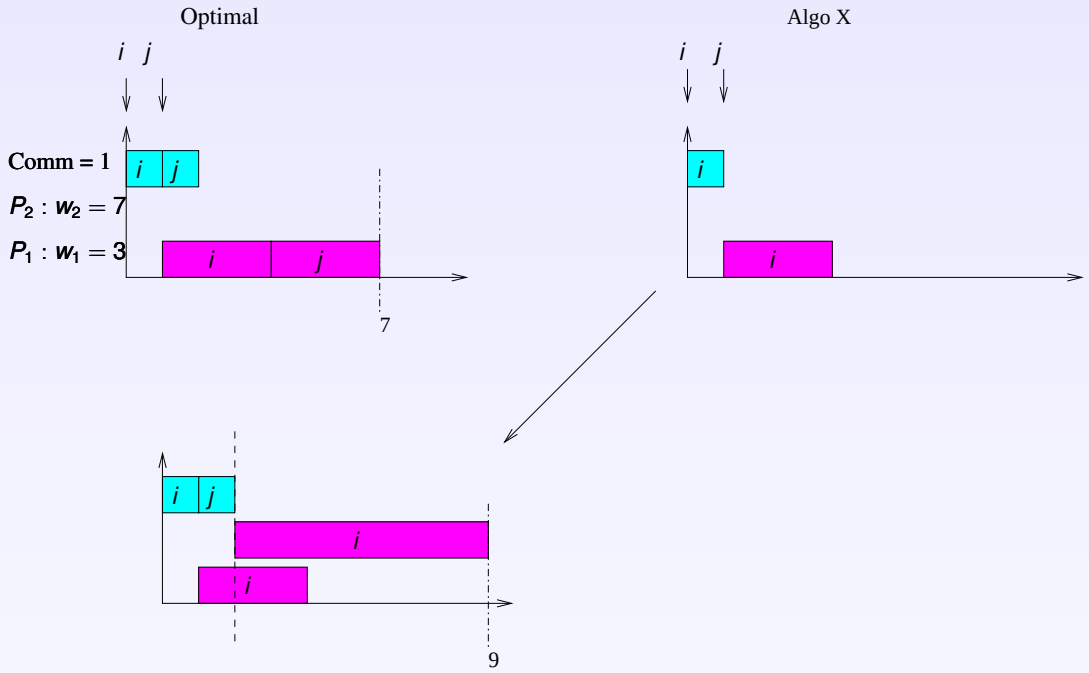
- $\mathcal{X}$ has no choice but to schedule task i on P_1 to enforce its competitive ratio.

Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

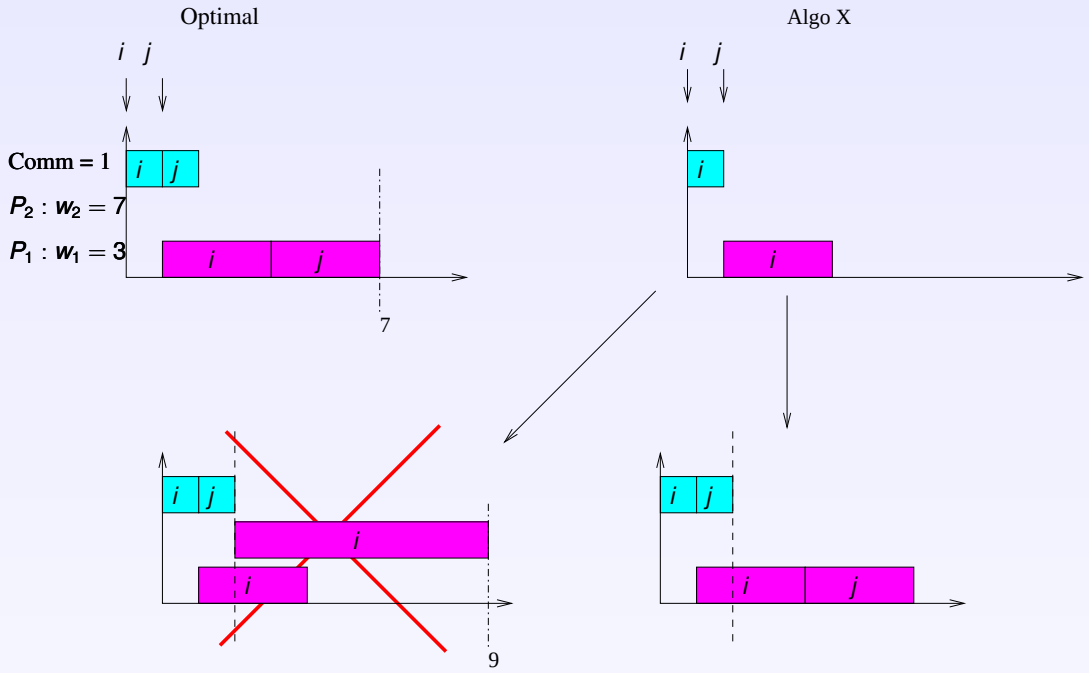
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- **competitive ratio :** $\frac{2c+w_2}{7} = \frac{9}{7} > \frac{5}{4} > \rho$.

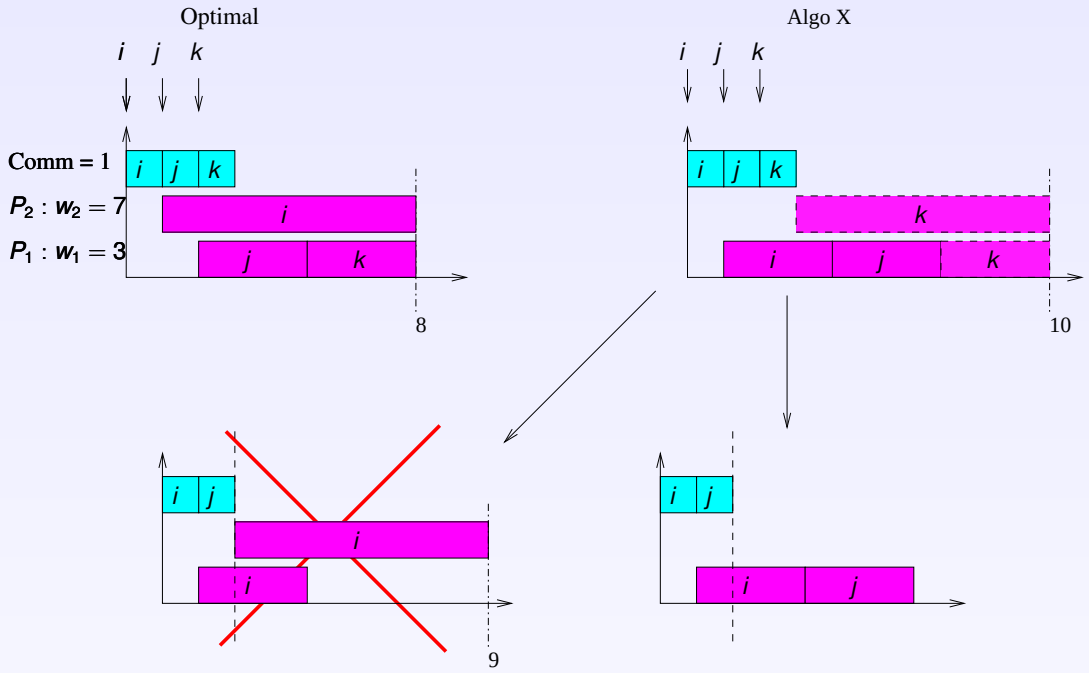
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- **competitive ratio :**

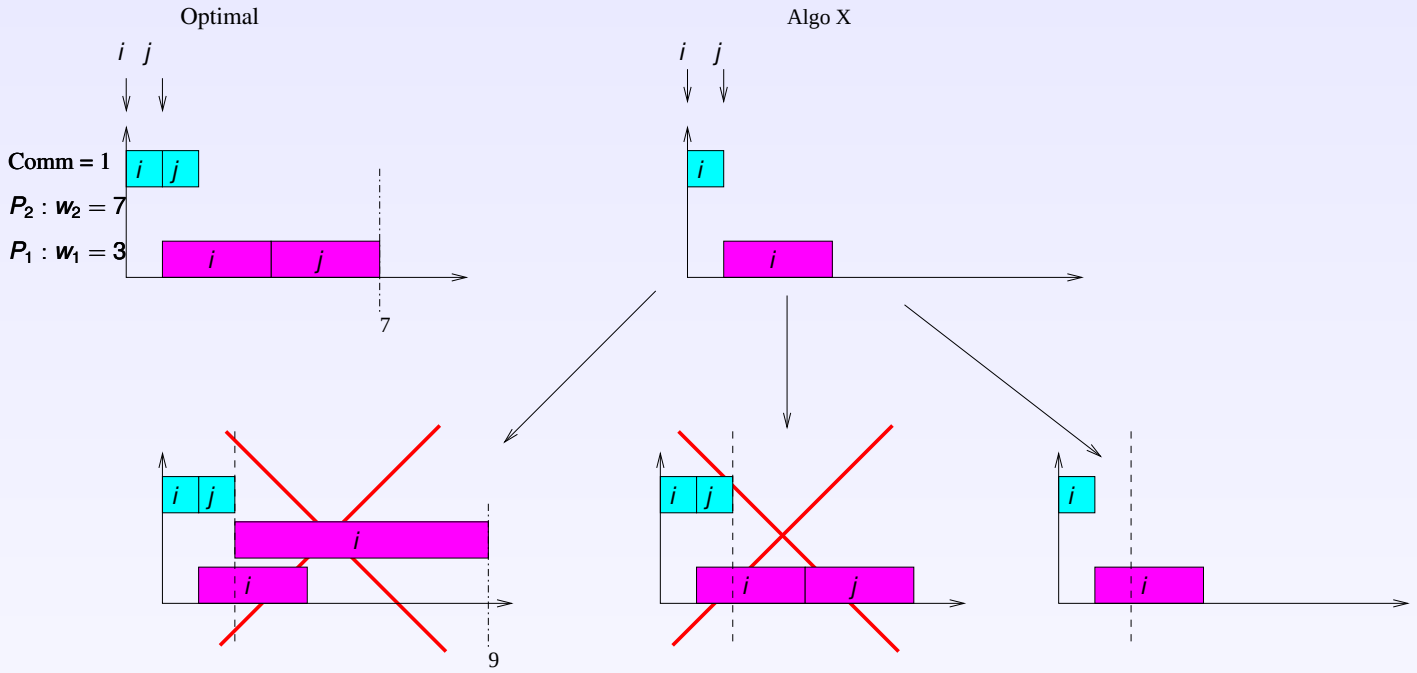
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- **competitive ratio :** $\frac{10}{8} = \frac{5}{4} > \rho$.

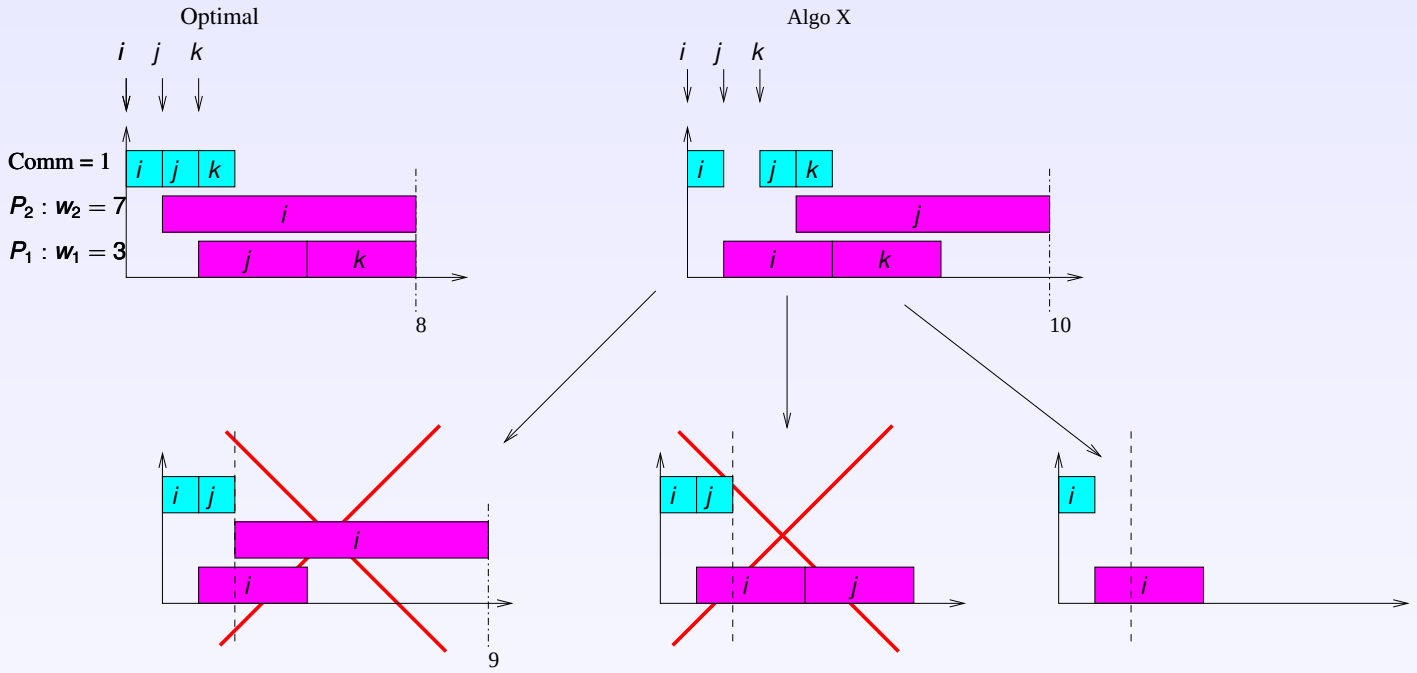
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- **competitive ratio :**

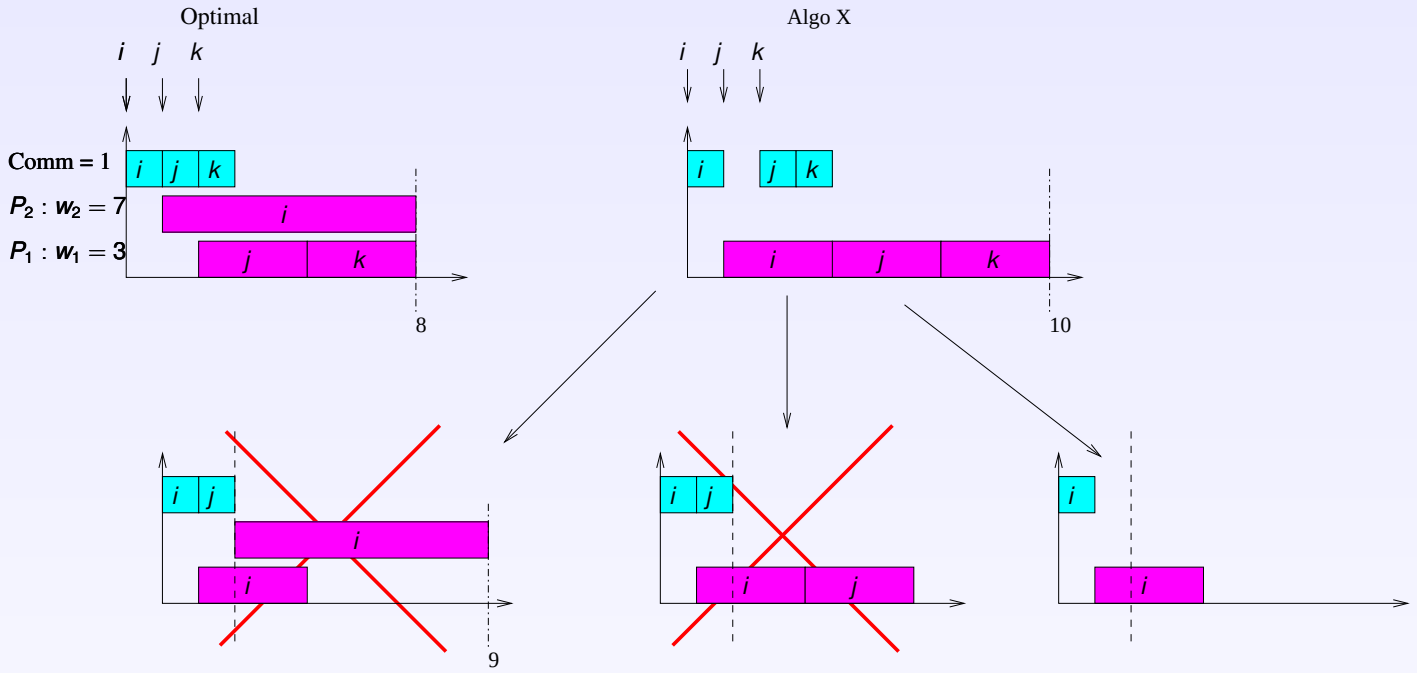
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- **competitive ratio :**

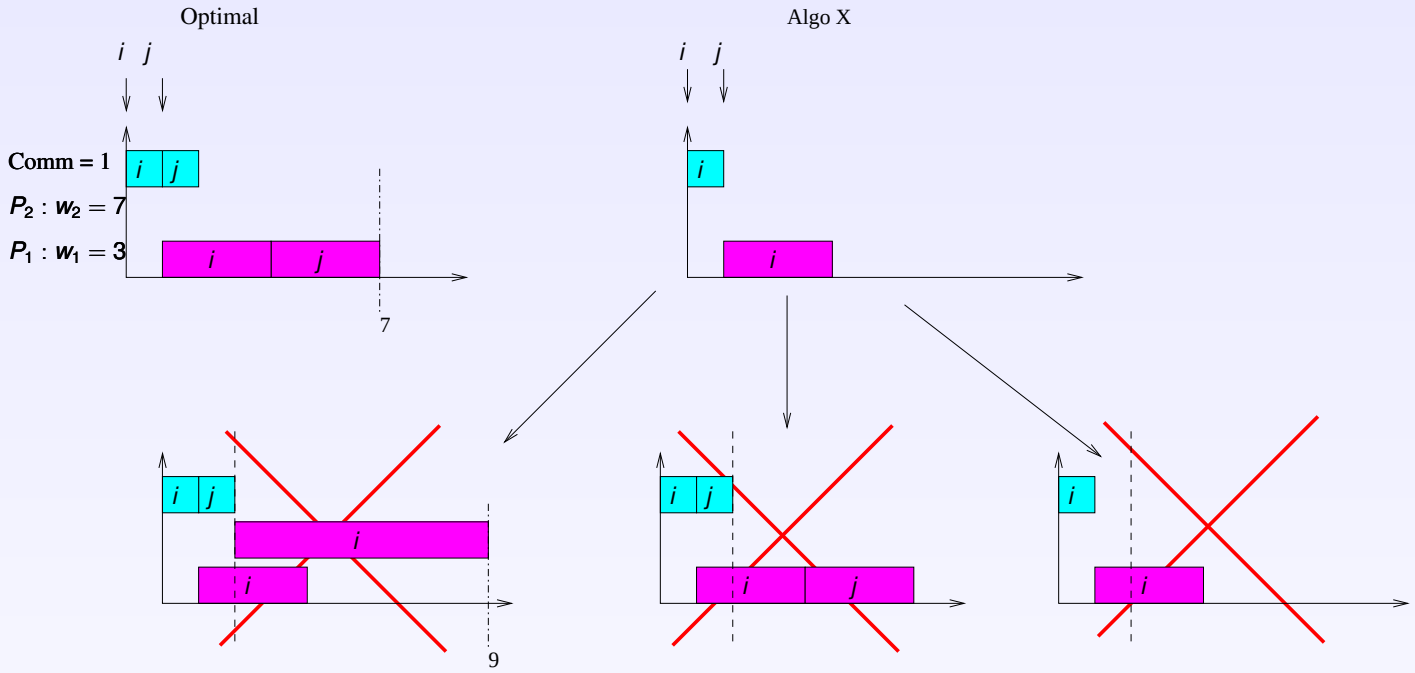
Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- **competitive ratio :**

Proof



At time $t_1 = c$, adversary sends task j. At time $t_2 = 2c$:

- **competitive ratio :** $\frac{10}{8} = \frac{5}{4} > \rho$.

Outline

General approach

How does it work?

Let's see how we find the worst platform for an on-line algorithm.

Example

- Fully heterogeneous platform
- Minimization of max flow

General approach

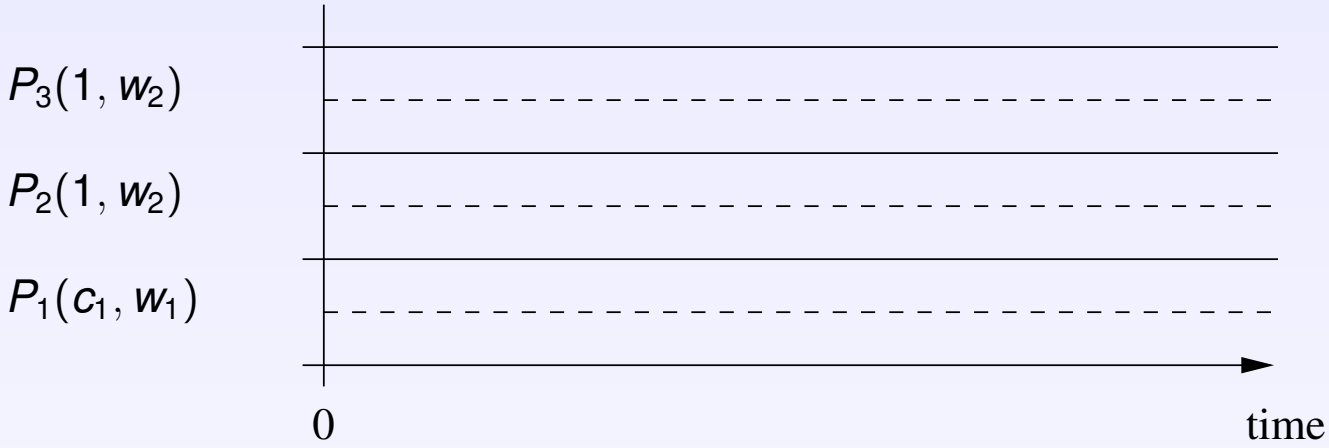
How does it work?

Let's see how we find the worst platform for an on-line algorithm.

Example

- Fully heterogeneous platform
- Minimization of max flow

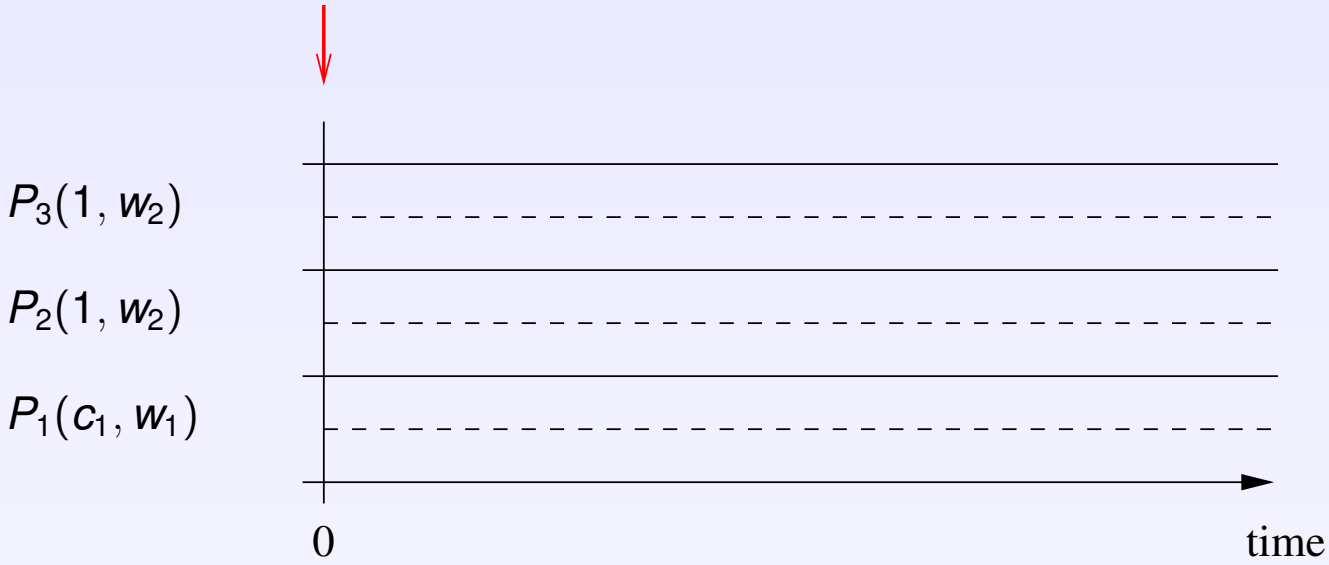
Generalisation



Idea:

- one fast processor with slow communication ($c_1 > 1$);
- two slow identical processors with fast communication;
- if only one task, send it on fast processor ($c_1 + w_1 < 1 + w_2$).
- if more than one task, do not send the first task on the fast processor

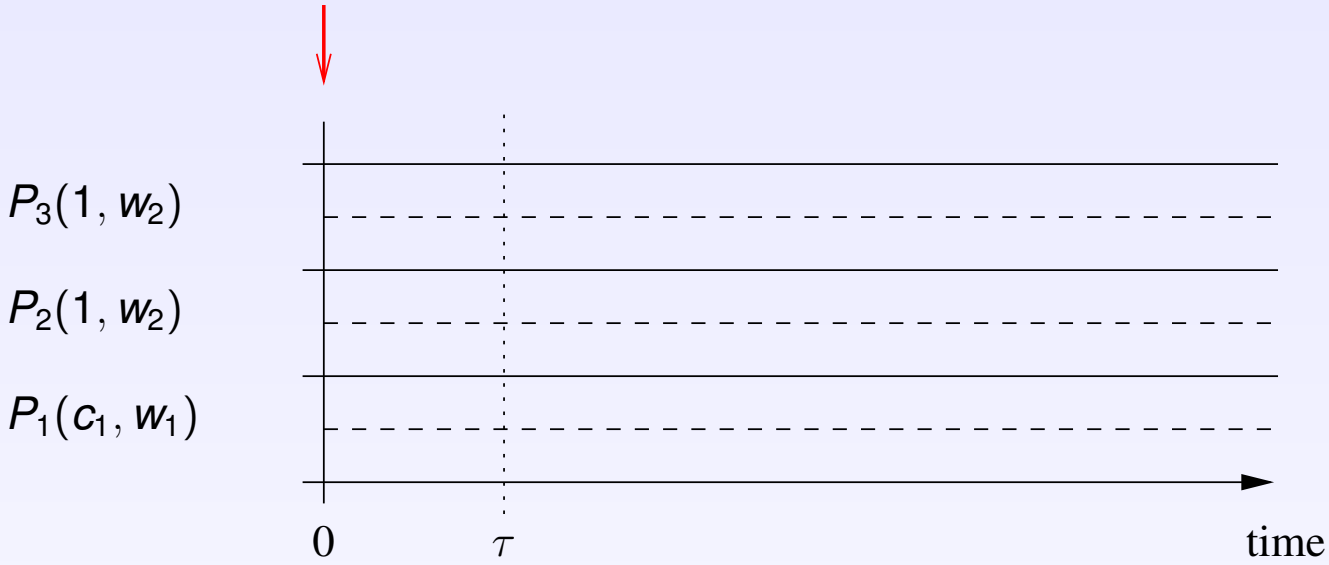
Generalisation



Idea:

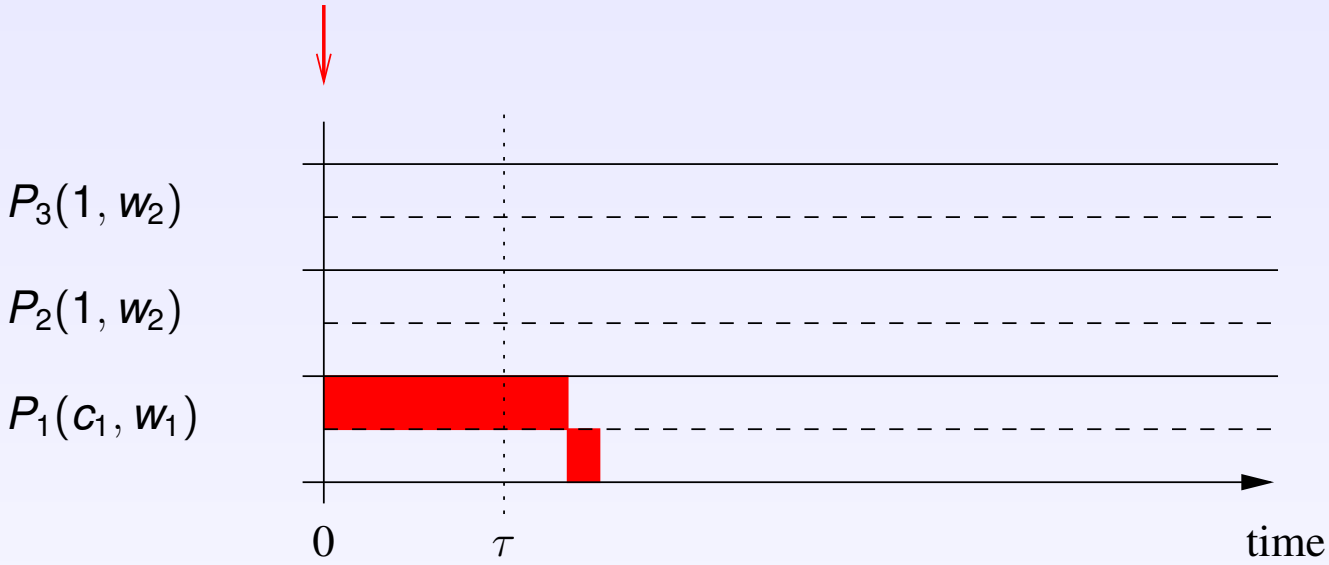
- one fast processor with slow communication ($c_1 > 1$);
- two slow identical processors with fast communication;
- if only one task, send it on fast processor ($c_1 + w_1 < 1 + w_2$).
- if more than one task, do not send the first task on the fast processor

Generalisation



At time $\tau \geq 1$ we look at what happened:

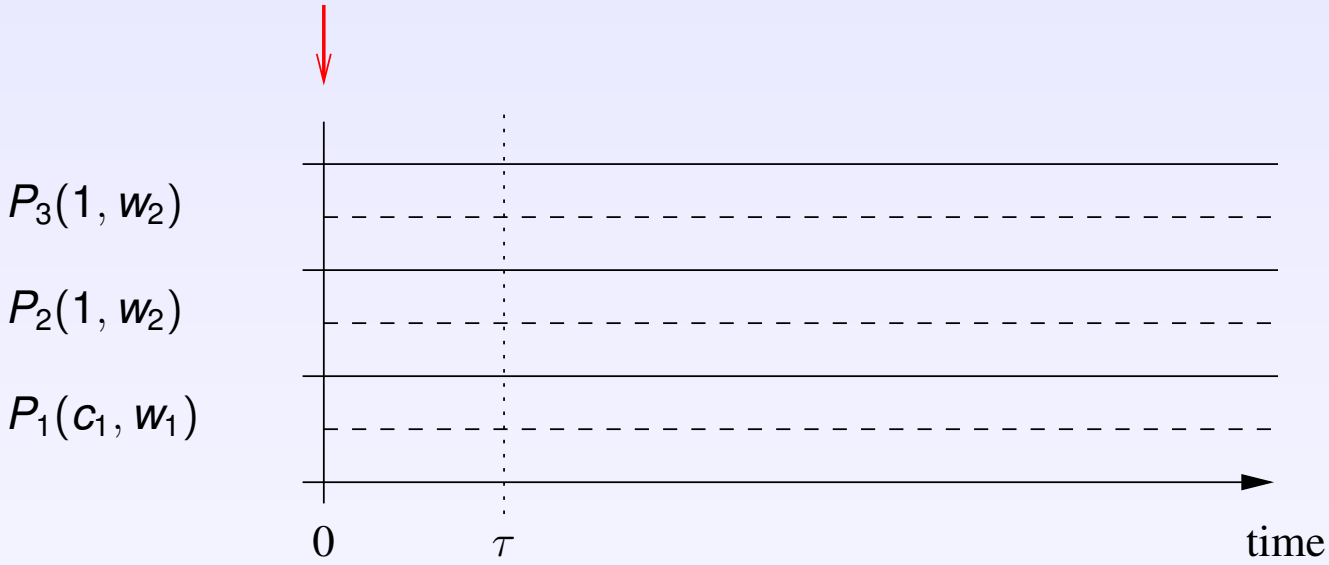
Generalisation



At time $\tau \geq 1$ we look at what happened:

1. Optimal : max flow $= c_1 + w_1$.

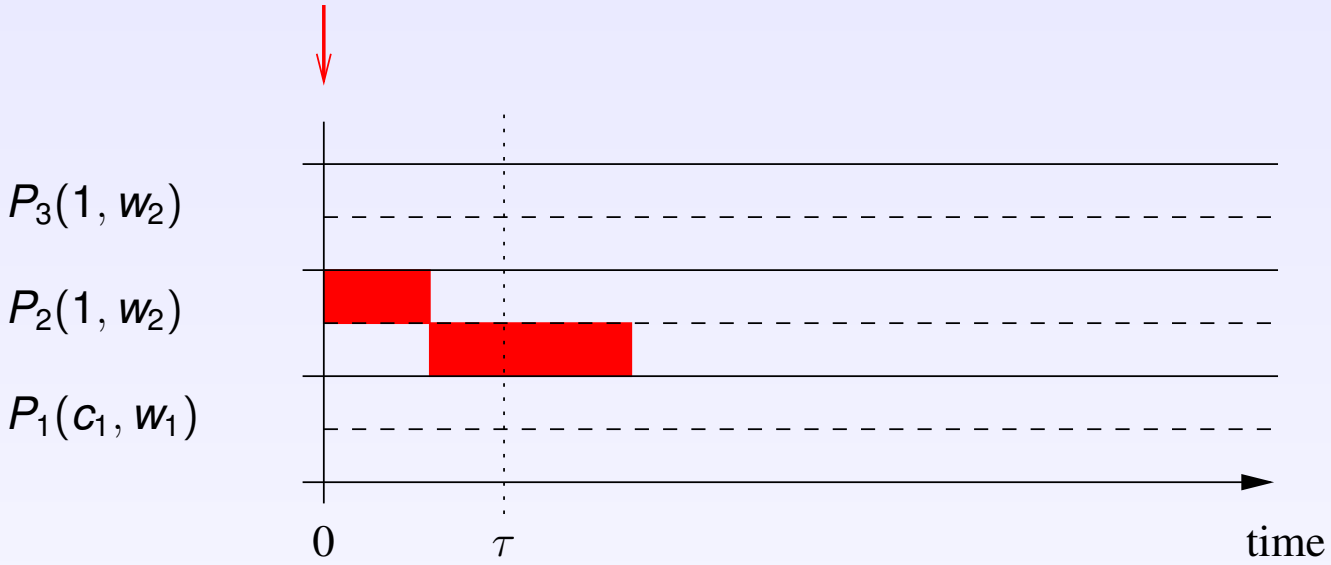
Generalisation



At time $\tau \geq 1$ we look at what happened:

1. Optimal : max flow $= c_1 + w_1$.
2. max flow $\geq \tau + c_1 + w_1$, ratio $\geq \frac{\tau + c_1 + w_1}{c_1 + w_1}$.

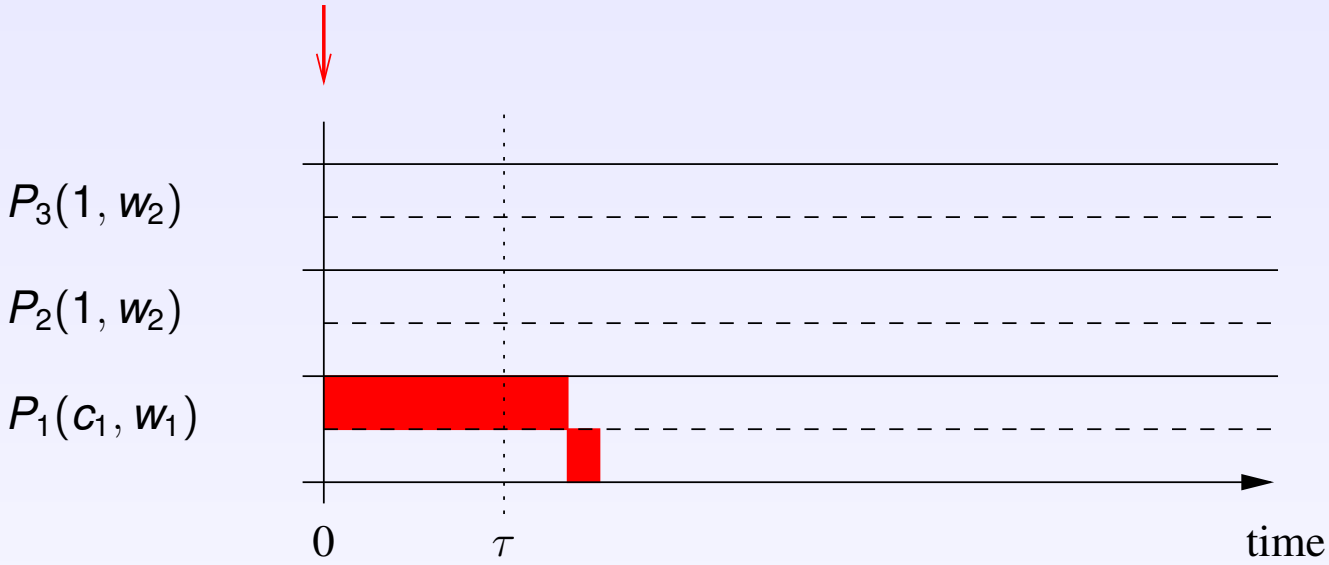
Generalisation



At time $\tau \geq 1$ we look at what happened:

1. Optimal : max flow $= c_1 + w_1$.
2. max flow $\geq \tau + c_1 + w_1$, ratio $\geq \frac{\tau + c_1 + w_1}{c_1 + w_1}$.
3. max flow $\geq 1 + w_2$, ratio $\geq \frac{1 + w_2}{c_1 + w_1}$.

Generalisation

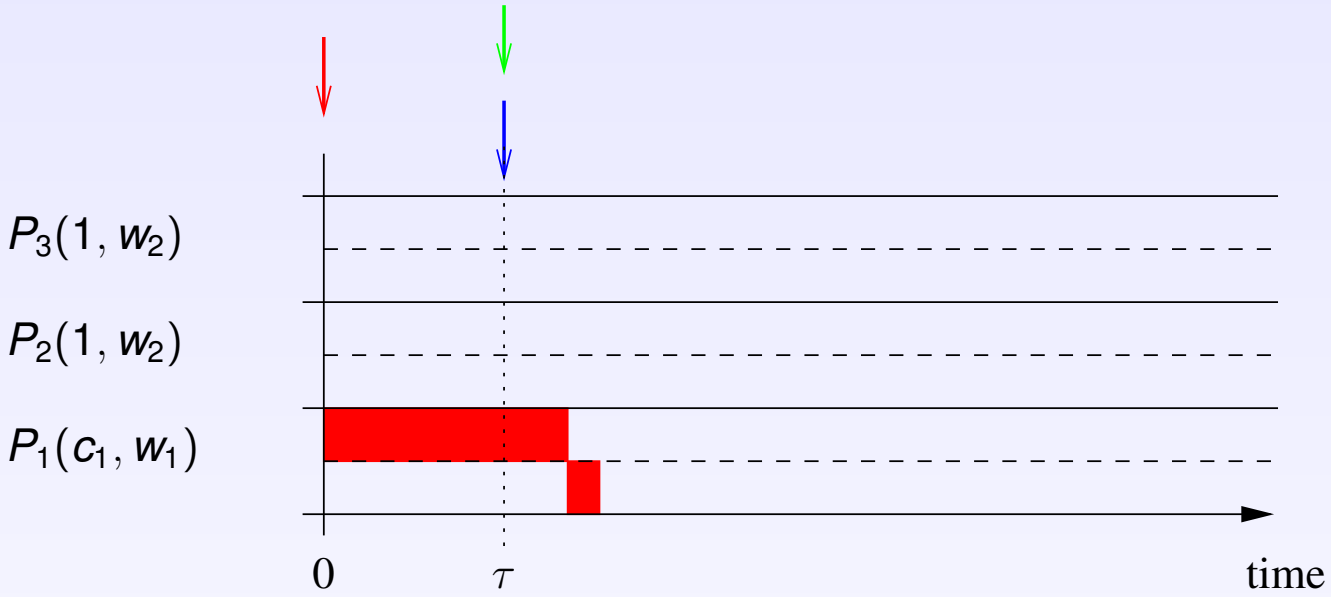


We choose τ, c_1, w_1 and w_2 to have:

$$\min \left\{ \frac{1 + w_2}{c_1 + w_1}, \frac{\tau + c_1 + w_1}{c_1 + w_1} \right\} \geq \rho$$

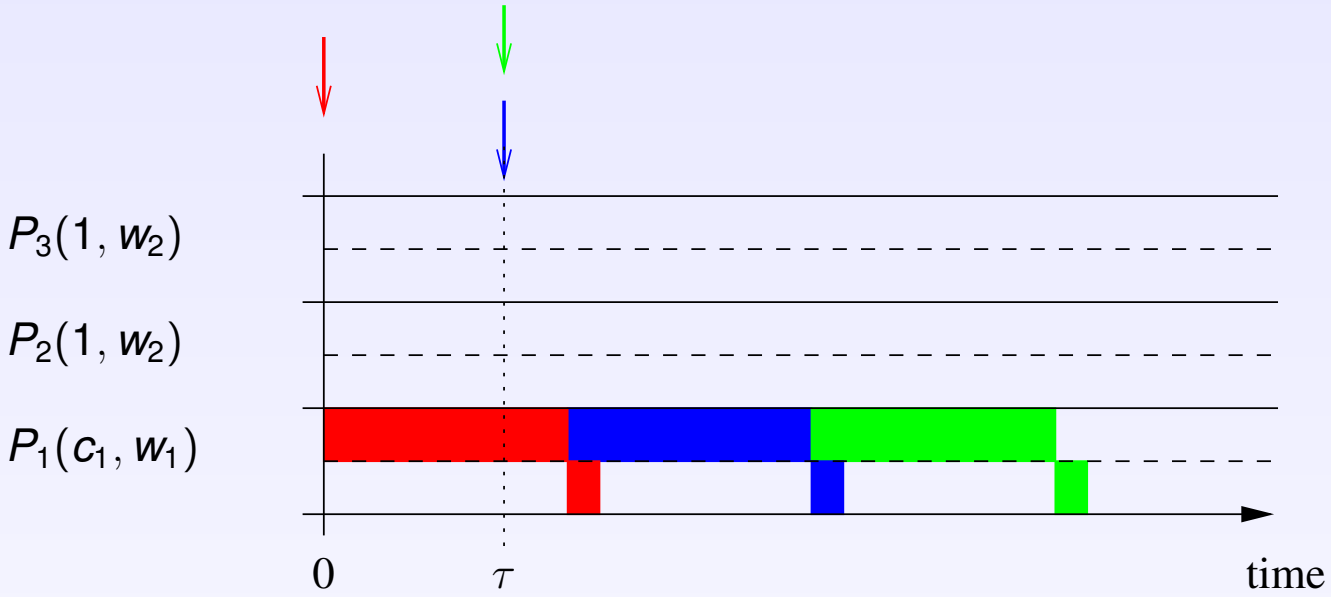
So algorithm has to execute the first task on P_1.

Generalisation



At time τ we send two new tasks. Let's see all possible schedulings.

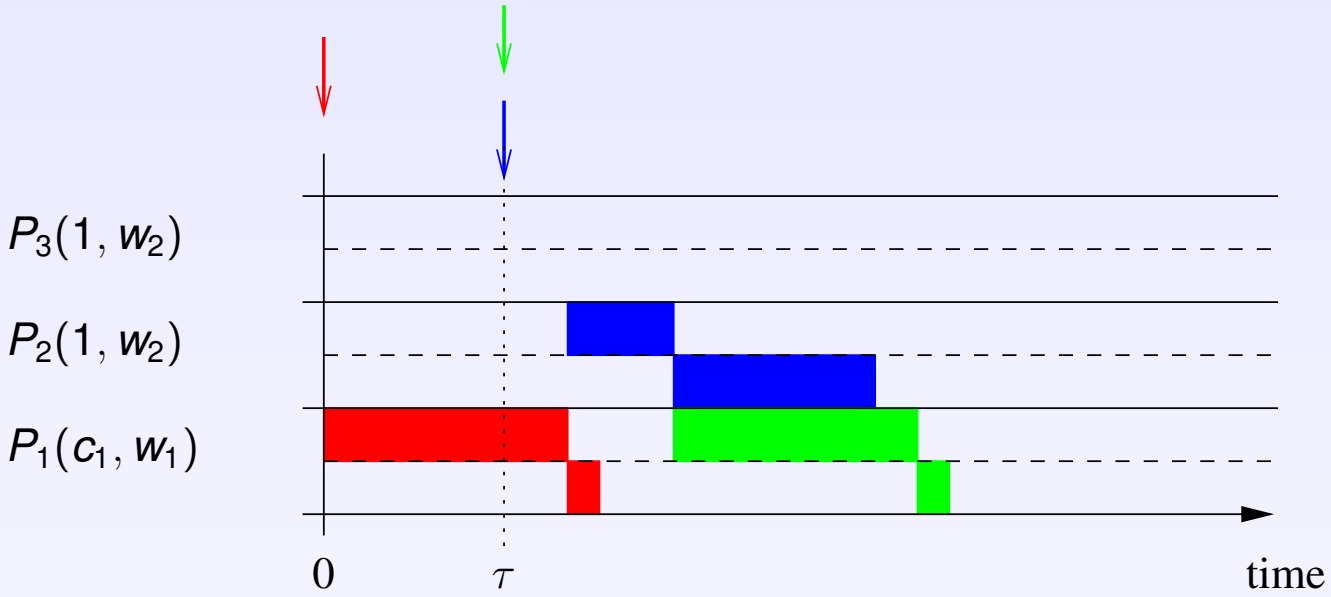
Generalisation



Max flow:

$$\max \begin{cases} c_1 + w_1, \\ \max\{\max\{c_1, \tau\} + c_1 + w_1, c_1 + 2w_1\} - \tau, \\ \max\{\max\{c_1, \tau\} + c_1 + w_1 + \max\{c_1, w_1\}, c_1 + 3w_1\} - \tau \end{cases}$$

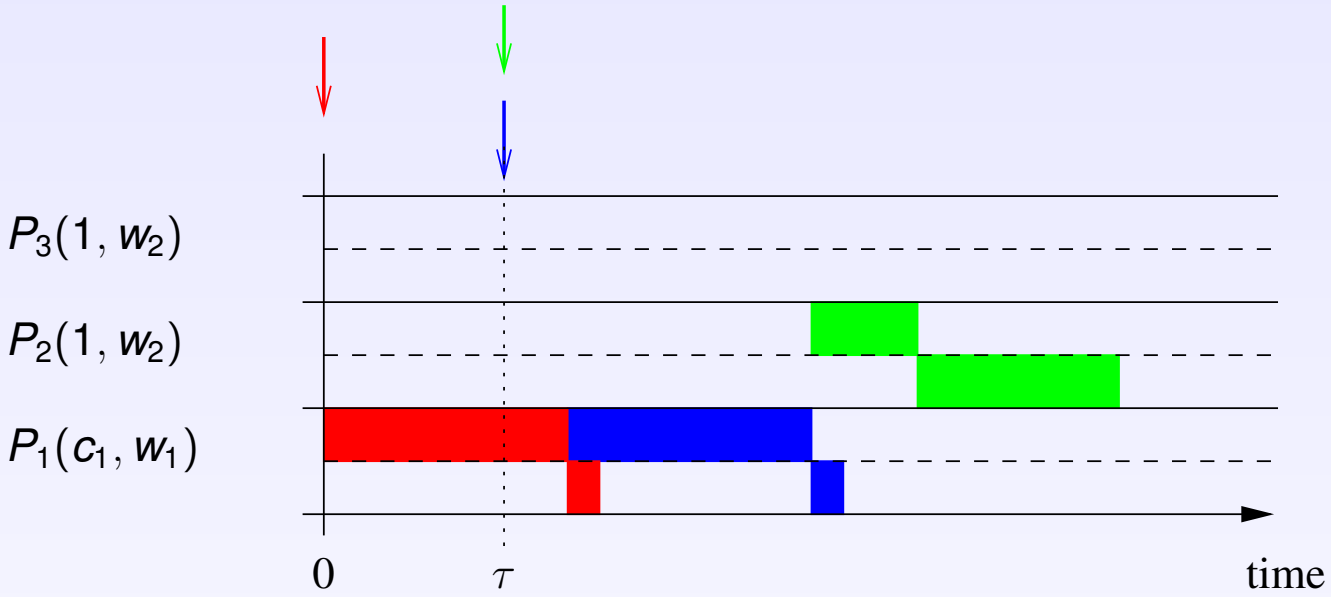
Generalisation



Max flow:

$$\max \begin{cases} c_1 + w_1, \\ (\max\{c_1, \tau\} + c_2 + w_2) - \tau, \\ \max\{\max\{c_1, \tau\} + c_2 + c_1 + w_1, c_1 + 2w_1\} - \tau \end{cases}$$

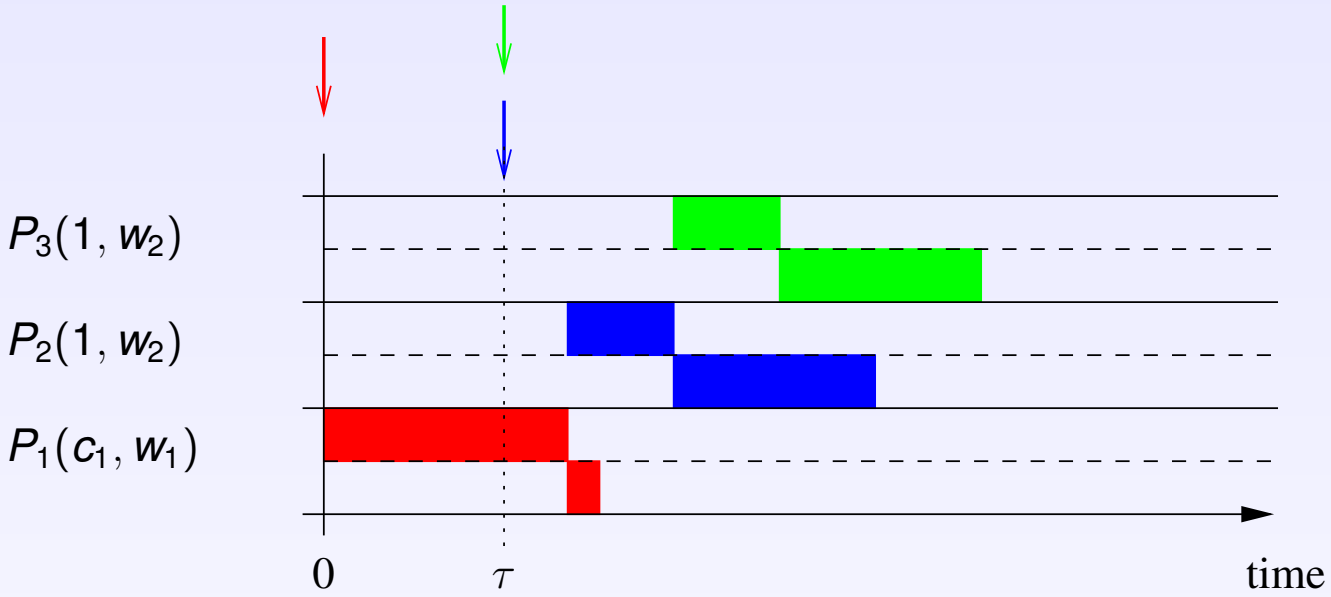
Generalisation



Max flow:

$$\max \begin{cases} c_1 + w_1, \\ \max\{\max\{c_1, \tau\} + c_1 + w_1, c_1 + 2w_1\} - \tau, \\ (\max\{c_1, \tau\} + c_1 + c_2 + w_2) - \tau\} \end{cases}$$

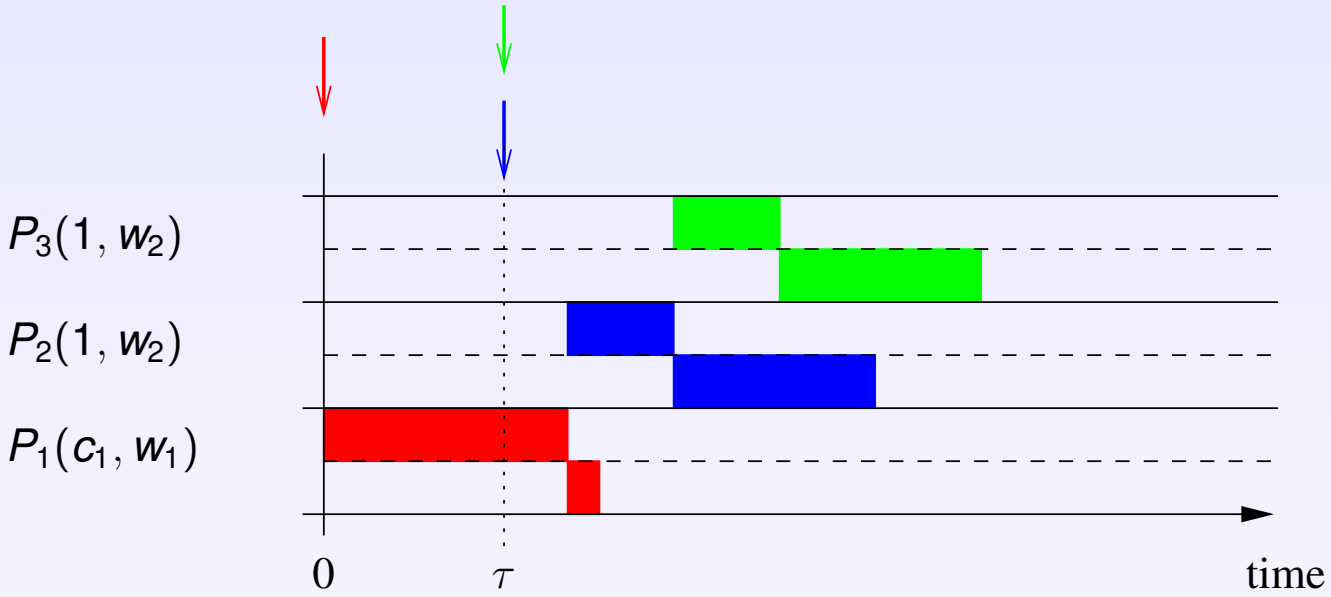
Generalisation



Max flow:

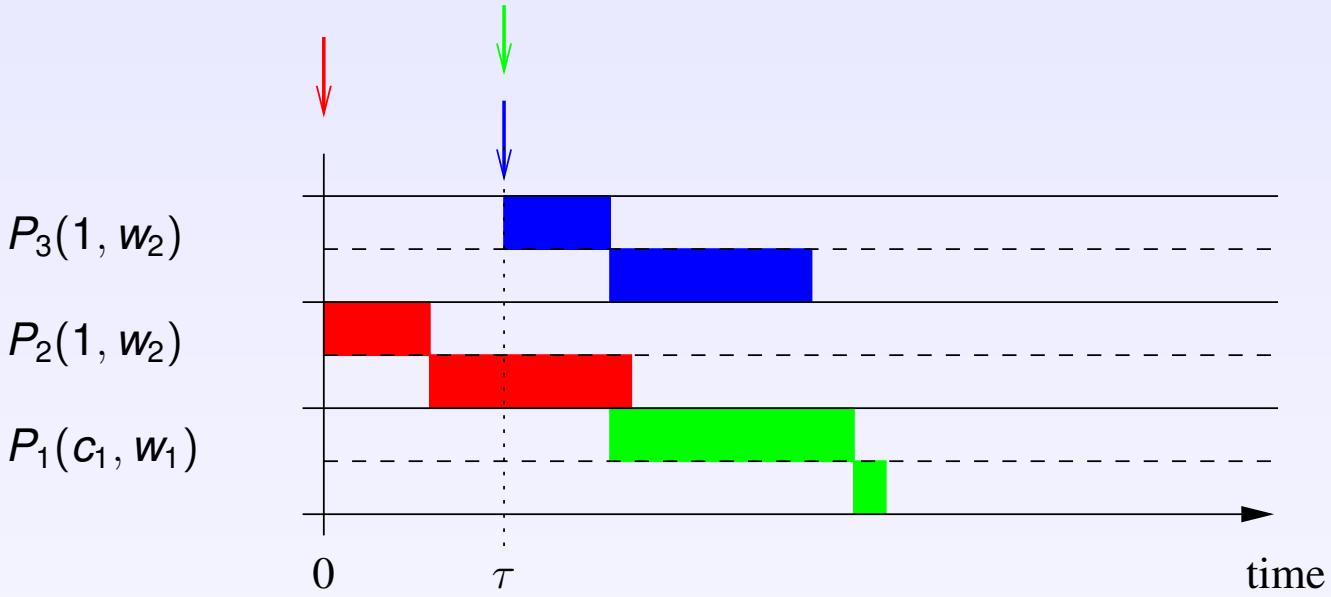
$$\max \begin{cases} c_1 + w_1, \\ (\max\{c_1, \tau\} + c_2 + w_2) - \tau, \\ (\max\{c_1, \tau\} + c_2 + c_2 + w_2) - \tau \end{cases}$$

Generalisation



The case where two tasks are allocated on P_2 is even worse than the previous case.

Generalisation



Better max flow:

$$\max \begin{cases} c_2 + w_2, \\ (\max\{c_2, \tau\} + c_2 + w_2) - \tau, \\ (\max\{c_2, \tau\} + c_2 + c_1 + w_1) - \tau \end{cases}$$

How we found lower bound of competitiveness (1)

Lower bound of competitiveness:

$$\min \left\{ \begin{array}{l} \frac{\tau+c_1+w_1}{c_1+w_1}, \\ \frac{1+w_2}{c_1+w_1}, \\ \min \left\{ \begin{array}{l} \max\{c_1+w_1, \max\{\max\{c_1,\tau\}+c_1+w_1, c_1+2w_1\}-\tau, \max\{\max\{c_1,\tau\}+c_1+w_1+\max\{c_1,w_1\}, c_1+3w_1\}-\tau\} \\ \max\{c_1+w_1, (\max\{c_1,\tau\}+c_2+w_2)-\tau, \max\{\max\{c_1,\tau\}+c_2+c_1+w_1, c_1+2w_1\}-\tau\} \\ \max\{c_1+w_1, \max\{\max\{c_1,\tau\}+c_1+w_1, c_1+2w_1\}-\tau, (\max\{c_1,\tau\}+c_1+c_2+w_2)-\tau\} \\ \max\{c_1+w_1, (\max\{c_1,\tau\}+c_2+w_2)-\tau, (\max\{c_1,\tau\}+c_2+c_2+w_2)-\tau\} \end{array} \right. \end{array} \right. \Big/ \max\{c_2+w_2, (\max\{c_2,\tau\}+c_2+c_1+w_1)-\tau\}$$

Problem

Find τ, c_1, w_1 and w_2 ($c_2 = 1$) which maximize this lower bound, such as : $c_1 + w_1 < 1 + w_2$.

How we found lower bound of competitiveness (1)

Lower bound of competitiveness:

$$\min \left\{ \begin{array}{l} \frac{\tau+c_1+w_1}{c_1+w_1}, \\ \frac{1+w_2}{c_1+w_1}, \\ \min \left\{ \begin{array}{l} \max\{c_1+w_1, \max\{\max\{c_1,\tau\}+c_1+w_1, c_1+2w_1\}-\tau, \max\{\max\{c_1,\tau\}+c_1+w_1+\max\{c_1,w_1\}, c_1+3w_1\}-\tau\} \\ \max\{c_1+w_1, (\max\{c_1,\tau\}+c_2+w_2)-\tau, \max\{\max\{c_1,\tau\}+c_2+c_1+w_1, c_1+2w_1\}-\tau\} \\ \max\{c_1+w_1, \max\{\max\{c_1,\tau\}+c_1+w_1, c_1+2w_1\}-\tau, (\max\{c_1,\tau\}+c_1+c_2+w_2)-\tau\} \\ \max\{c_1+w_1, (\max\{c_1,\tau\}+c_2+w_2)-\tau, (\max\{c_1,\tau\}+c_2+c_2+w_2)-\tau\} \end{array} \right. \end{array} \right. \Big/ \max\{c_2+w_2, (\max\{c_2,\tau\}+c_2+c_1+w_1)-\tau\}$$

Problem

Find τ, c_1, w_1 and w_2 ($c_2 = 1$) which maximize this lower bound, such as : $c_1 + w_1 < 1 + w_2$.

How we found lower bound of competitiveness (2)

1. Numerical resolution
2. Characterization of optimal : $\tau < c_1$, $w_1 = 0$, etc.
3. New system:

$$\min \left\{ \begin{array}{l} \frac{\tau+c_1}{c_1}, \\ \frac{1+w_2}{c_1}, \\ \frac{\min \left\{ \begin{array}{l} 3c_1 - \tau, \\ c_1 + 1 - \tau + w_2, \\ 2c_1 - \tau + 1 + w_2 \\ c_1 + 2 + w_2 - \tau \end{array} \right.}{1+w_2} \end{array} \right. = \min \left\{ \begin{array}{l} \frac{\tau+c_1}{c_1}, \\ \frac{1+w_2}{c_1}, \\ \frac{c_1+1-\tau+w_2}{1+w_2} \end{array} \right.$$

4. Solution: $c_1 = 2(1+\sqrt{2})$, $w_2 = \sqrt{2}c_1 - 1$, $\tau = 2$, $\rho = \sqrt{2}$.

How we found lower bound of competitiveness (2)

1. Numerical resolution
2. Characterization of optimal : $\tau < c_1$, $w_1 = 0$, etc.
3. New system:

$$\min \left\{ \begin{array}{l} \frac{\tau+c_1}{c_1}, \\ \frac{1+w_2}{c_1}, \\ \frac{\min \left\{ \begin{array}{l} 3c_1 - \tau, \\ c_1 + 1 - \tau + w_2, \\ 2c_1 - \tau + 1 + w_2 \\ c_1 + 2 + w_2 - \tau \end{array} \right.}{1+w_2} \end{array} \right. = \min \left\{ \begin{array}{l} \frac{\tau+c_1}{c_1}, \\ \frac{1+w_2}{c_1}, \\ \frac{c_1+1-\tau+w_2}{1+w_2} \end{array} \right.$$

4. Solution: $c_1 = 2(1+\sqrt{2})$, $w_2 = \sqrt{2}c_1 - 1$, $\tau = 2$, $\rho = \sqrt{2}$.

How we found lower bound of competitiveness (2)

1. Numerical resolution
2. Characterization of optimal : $\tau < c_1$, $w_1 = 0$, etc.
3. New system:

$$\min \left\{ \begin{array}{l} \frac{\tau + c_1}{c_1}, \\ \frac{1 + w_2}{c_1}, \\ \dfrac{\min \left\{ \begin{array}{l} 3c_1 - \tau, \\ c_1 + 1 - \tau + w_2, \\ 2c_1 - \tau + 1 + w_2 \\ c_1 + 2 + w_2 - \tau \end{array} \right.}{1 + w_2} \end{array} \right. = \min \left\{ \begin{array}{l} \frac{\tau + c_1}{c_1}, \\ \frac{1 + w_2}{c_1}, \\ \frac{c_1 + 1 - \tau + w_2}{1 + w_2}, \end{array} \right.$$

4. Solution: $c_1 = 2(1 + \sqrt{2})$, $w_2 = \sqrt{2}c_1 - 1$, $\tau = 2$, $\rho = \sqrt{2}$.

How we found lower bound of competitiveness (2)

1. Numerical resolution
2. Characterization of optimal : $\tau < c_1$, $w_1 = 0$, etc.
3. New system:

$$\min \left\{ \begin{array}{l} \frac{\tau + c_1}{c_1}, \\ \frac{1 + w_2}{c_1}, \\ \dfrac{\min \left\{ \begin{array}{l} 3c_1 - \tau, \\ c_1 + 1 - \tau + w_2, \\ 2c_1 - \tau + 1 + w_2 \\ c_1 + 2 + w_2 - \tau \end{array} \right.}{1 + w_2} \end{array} \right. = \min \left\{ \begin{array}{l} \frac{\tau + c_1}{c_1}, \\ \frac{1 + w_2}{c_1}, \\ \frac{c_1 + 1 - \tau + w_2}{1 + w_2}, \end{array} \right.$$

4. Solution: $c_1 = 2(1 + \sqrt{2})$, $w_2 = \sqrt{2}c_1 - 1$, $\tau = 2$, $\rho = \sqrt{2}$.

Outline

All results

Platform type	Objective function		
	Makespan	Max-flow	Sum-flow
Homogeneous	1	1	1
Communication homogeneous (with more than **two** slaves)	$\frac{5}{4} = 1.250$	$\frac{5-\sqrt{7}}{2} \approx 1.177$	$\frac{2+4\sqrt{2}}{7} \approx 1.093$
Computation homogeneous (with more than **two** slaves)	$\frac{6}{5} = 1.200$	$\frac{5}{4} = 1.250$	$\frac{23}{22} \approx 1.045$
Heterogeneous (with more than **three** slaves)	$\frac{1+\sqrt{3}}{2} \approx 1.366$	$\sqrt{2} \approx 1.414$	$\frac{\sqrt{13}-1}{2} \approx 1.302$

Table: Lower bounds on the competitive ratio of on-line algorithms, depending on the platform type and on the objective function.

Outline

Outline

Scheduling the Last Jobs First

Optimal algorithm...

to minimize the *makespan* as soon as it knows the total number of tasks.

Scheduling the Last Jobs First

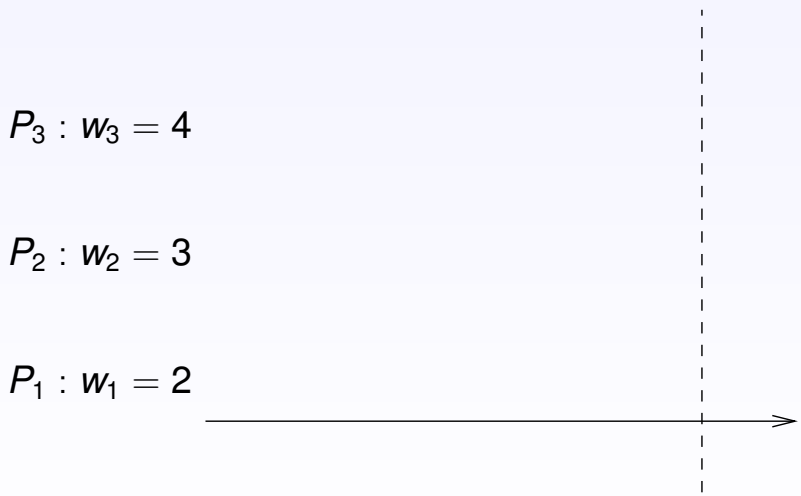
How does it work?

- Set a virtual deadline
- Schedule the tasks as late as possible, according to previous choices
- Memorize the scheduling of the tasks to the processors
- Send them as soon as possible

Scheduling the Last Jobs First

How does it work?

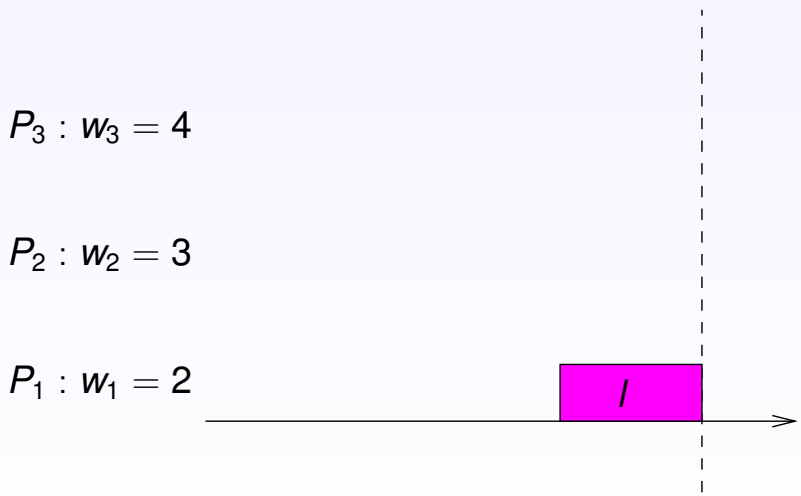
- Set a virtual deadline
- Schedule the tasks as late as possible, according to previous choices
- Memorize the scheduling of the tasks to the processors
- Send them as soon as possible



Scheduling the Last Jobs First

How does it work?

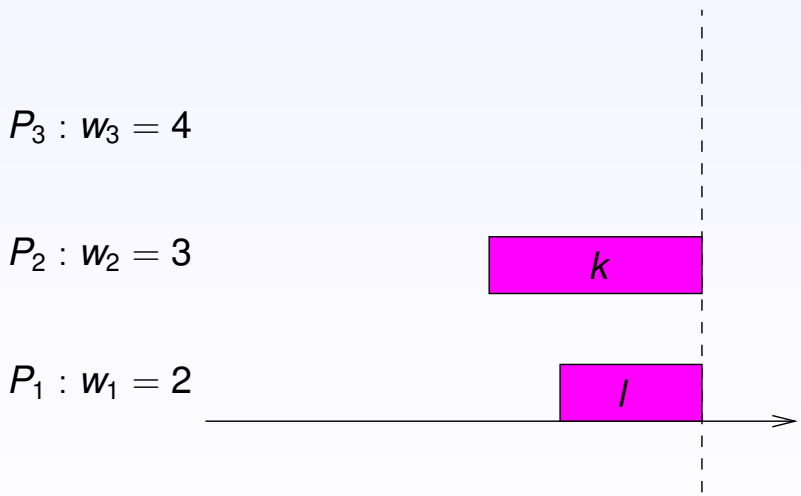
- Set a virtual deadline
- Schedule the tasks as late as possible, according to previous choices
- Memorize the scheduling of the tasks to the processors
- Send them as soon as possible



Scheduling the Last Jobs First

How does it work?

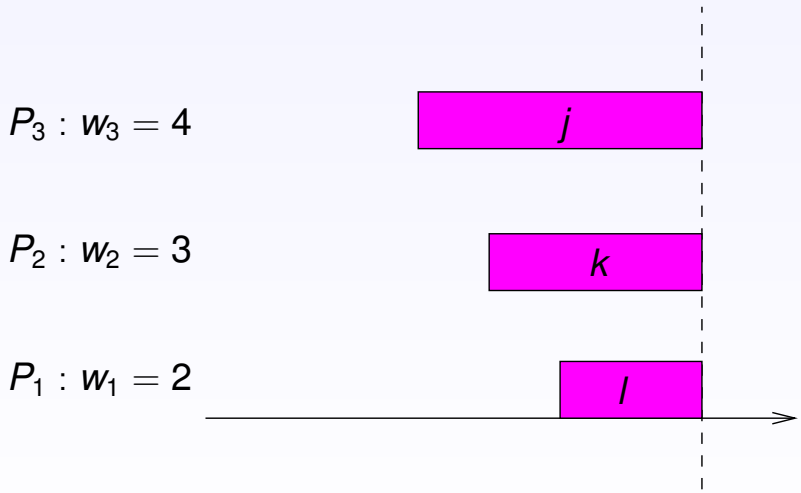
- Set a virtual deadline
- Schedule the tasks as late as possible, according to previous choices
- Memorize the scheduling of the tasks to the processors
- Send them as soon as possible



Scheduling the Last Jobs First

How does it work?

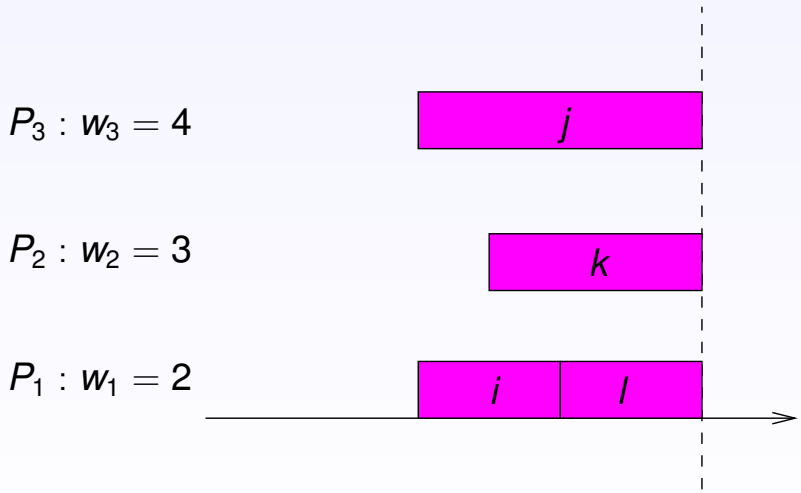
- Set a virtual deadline
- Schedule the tasks as late as possible, according to previous choices
- Memorize the scheduling of the tasks to the processors
- Send them as soon as possible



Scheduling the Last Jobs First

How does it work?

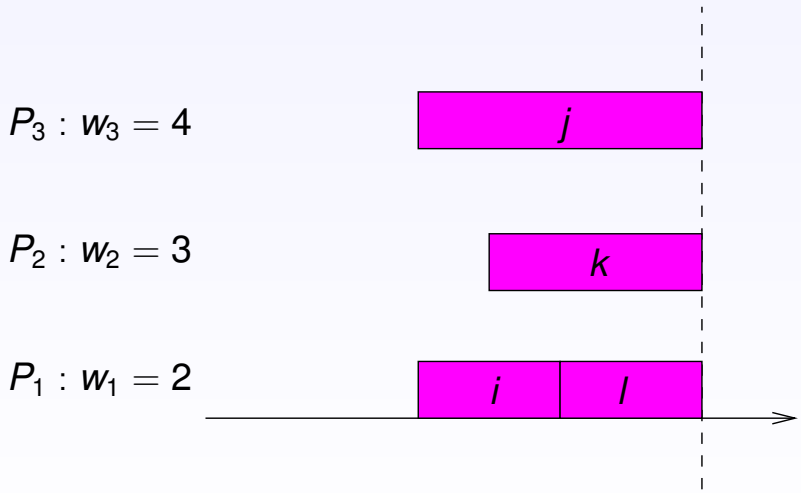
- Set a virtual deadline
- Schedule the tasks as late as possible, according to previous choices
- Memorize the scheduling of the tasks to the processors
- Send them as soon as possible



Scheduling the Last Jobs First

How does it work?

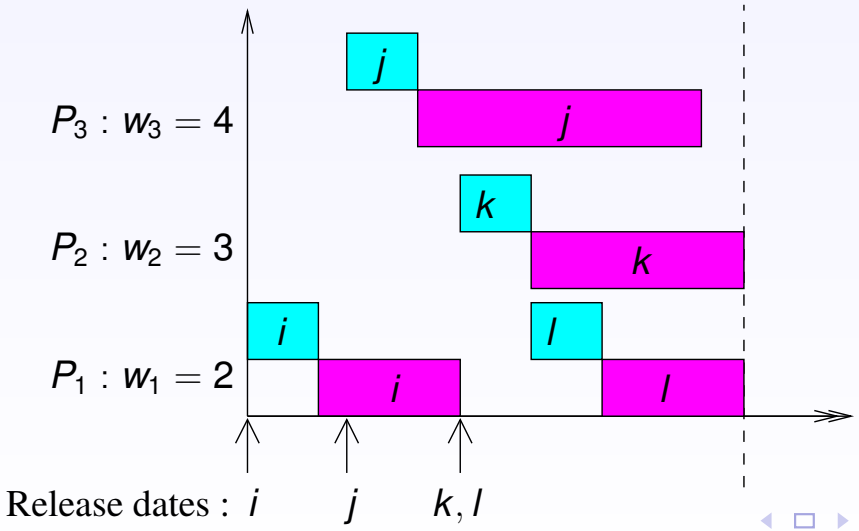
- Set a virtual deadline
- Schedule the tasks as late as possible, according to previous choices
- Memorize the scheduling of the tasks to the processors
- Send them as soon as possible



Scheduling the Last Jobs First

How does it work?

- Set a virtual deadline
- Schedule the tasks as late as possible, according to previous choices
- Memorize the scheduling of the tasks to the processors
- Send them as soon as possible



Outline

Sometimes *Round-Robin* is optimal

If

$$\sum_{i=1}^{m} c_i \leq w$$

Round-Robin is the optimal scheduling to minimize the *makespan*.

Sometimes *Round-Robin* is optimal

If

$$\sum_{i=1}^{m} c_i \leq w$$

Round-Robin is the optimal scheduling to minimize the *makespan*.

$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 15$

Sometimes *Round-Robin* is optimal

If

$$\sum_{i=1}^{m} c_i \leq w$$

Round-Robin is the optimal scheduling to minimize the *makespan*.

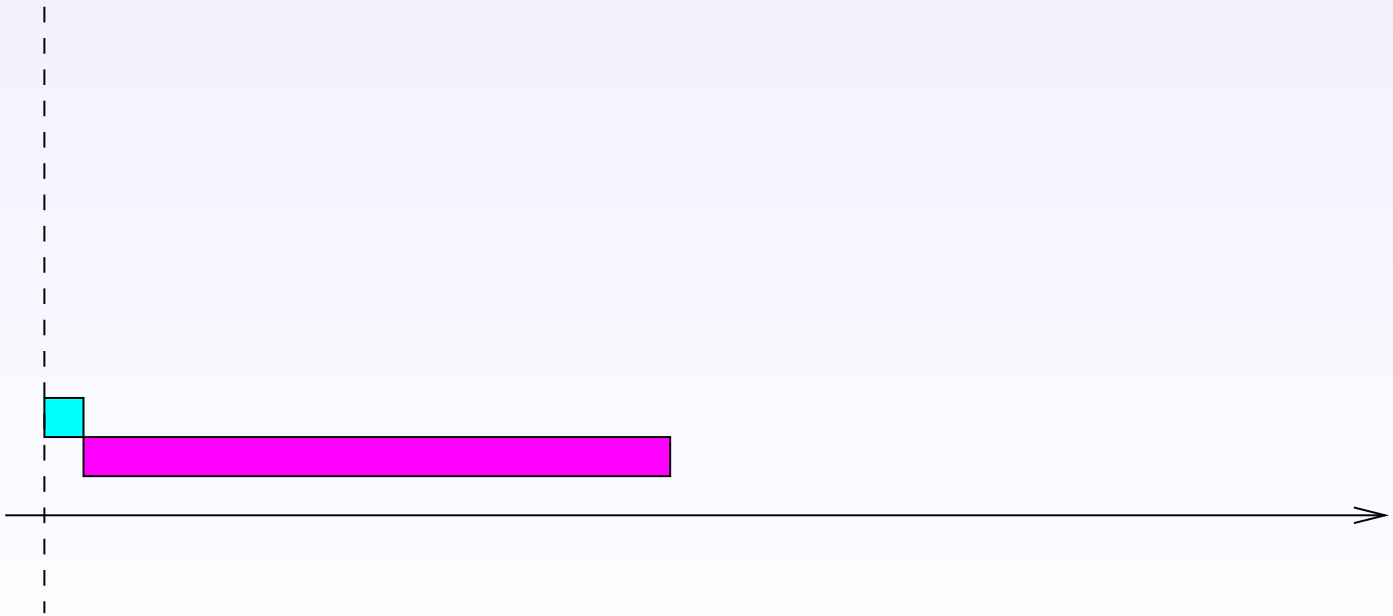
$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 15$



Sometimes *Round-Robin* is optimal

If

$$\sum_{i=1}^{m} c_i \leq w$$

Round-Robin is the optimal scheduling to minimize the *makespan*.

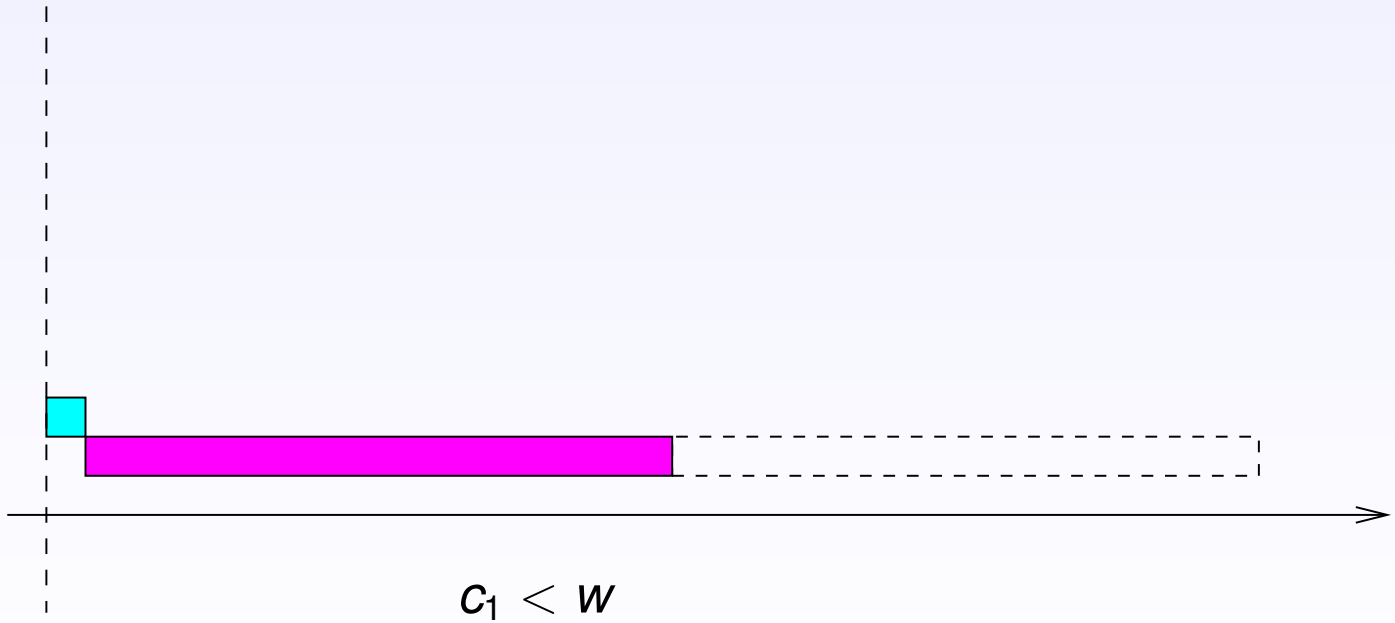
$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 15$

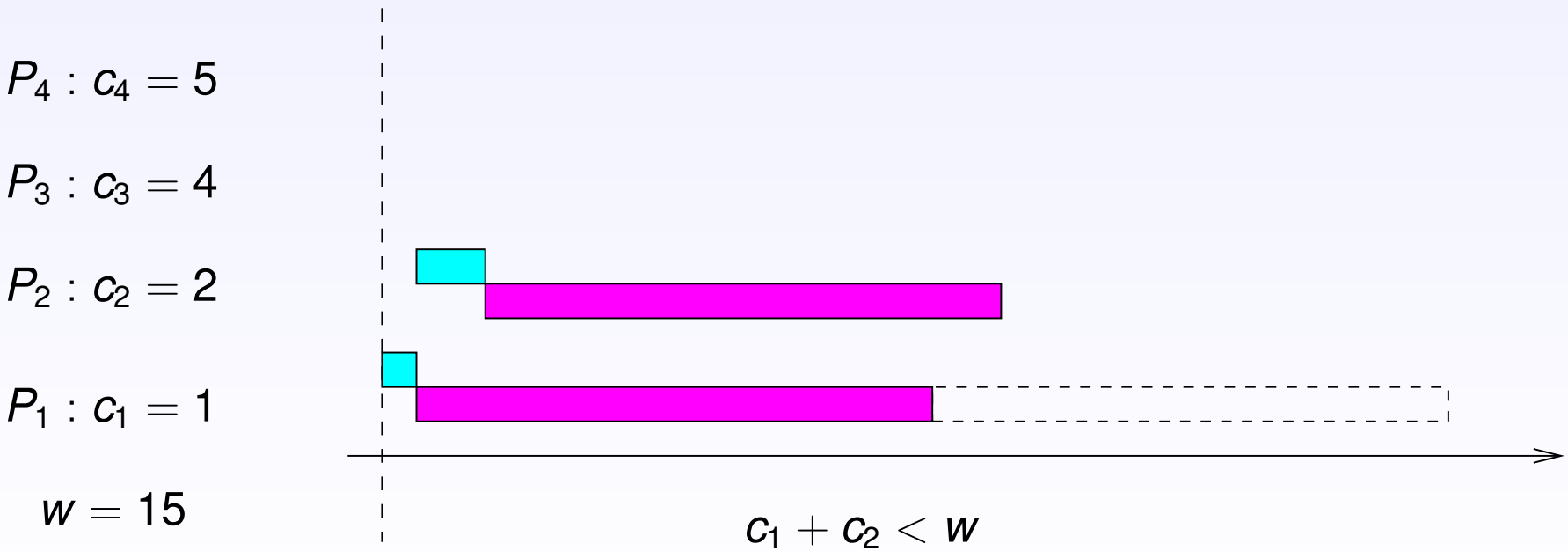


Sometimes *Round-Robin* is optimal

If

$$\sum_{i=1}^{m} c_i \leq w$$

Round-Robin is the optimal scheduling to minimize the *makespan*.

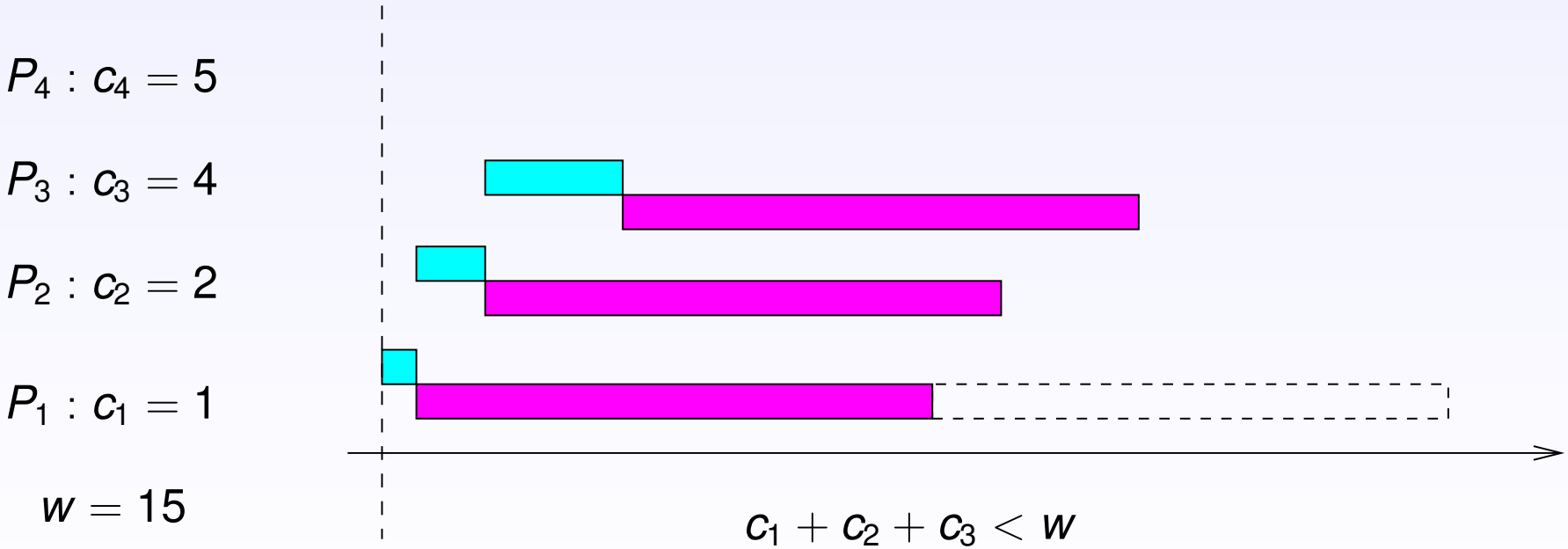


Sometimes *Round-Robin* is optimal

If

$$\sum_{i=1}^{m} c_i \leq w$$

Round-Robin is the optimal scheduling to minimize the *makespan*.

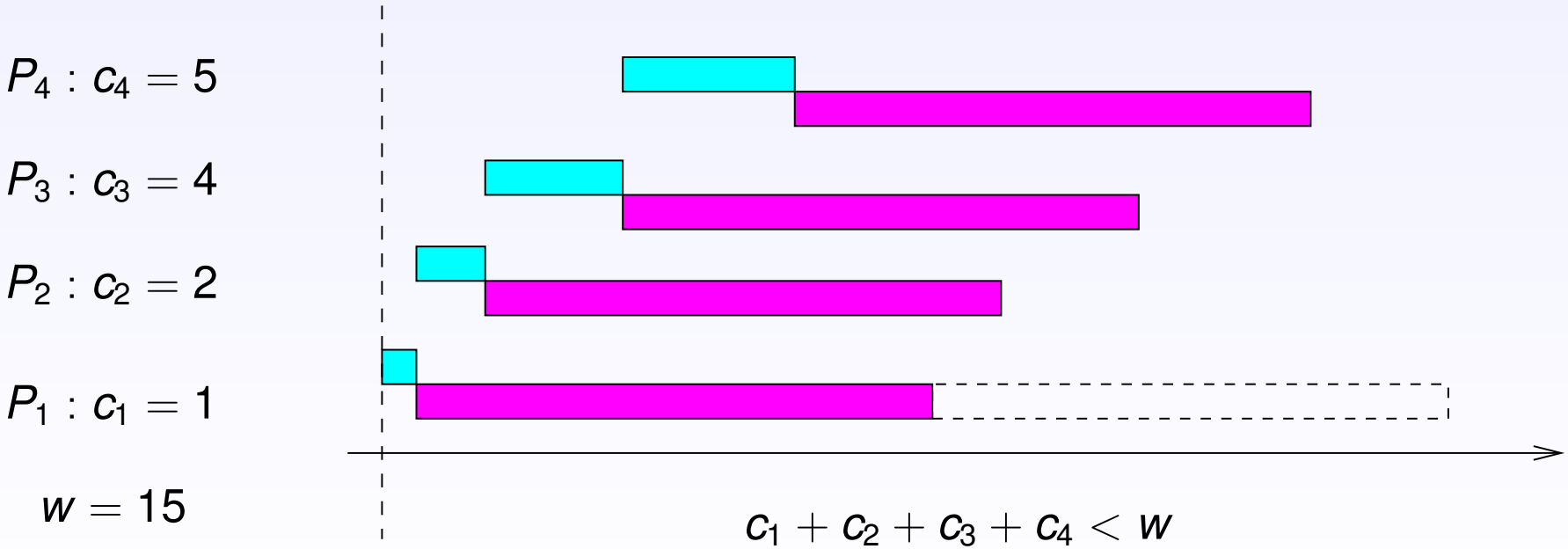


Sometimes *Round-Robin* is optimal

If

$$\sum_{i=1}^{m} c_i \leq w$$

Round-Robin is the optimal scheduling to minimize the *makespan*.

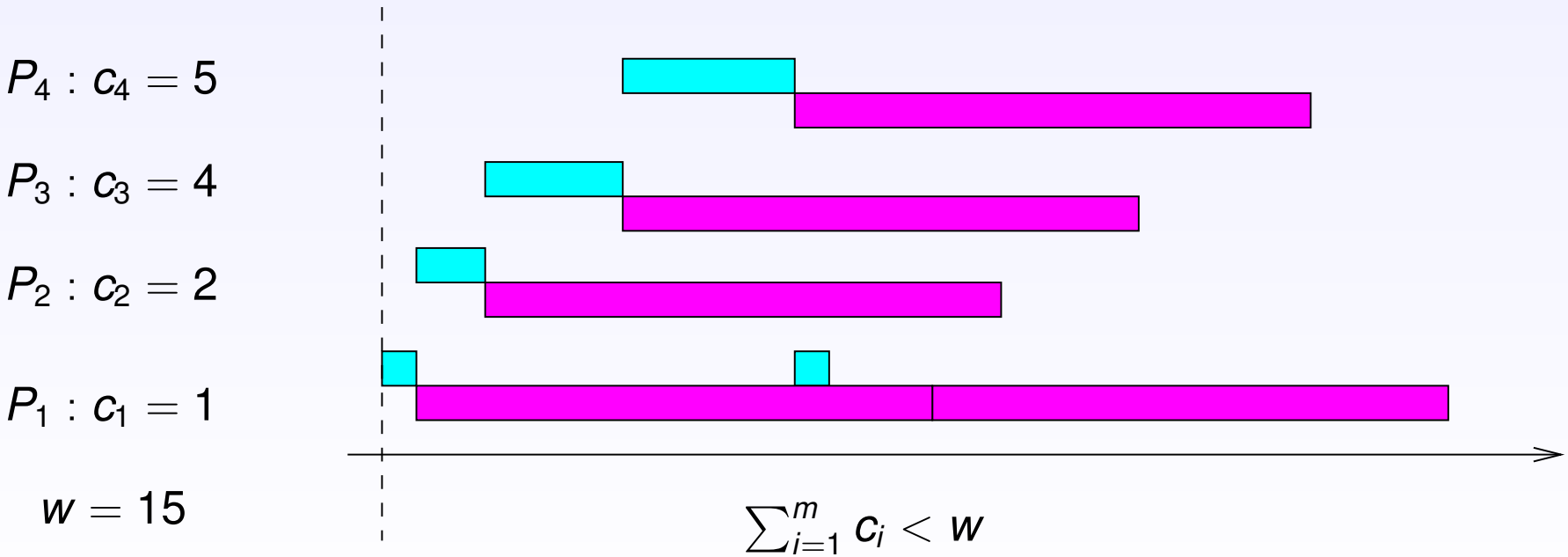


Sometimes *Round-Robin* is optimal

If

$$\sum_{i=1}^{m} c_i \leq w$$

Round-Robin is the optimal scheduling to minimize the *makespan*.



Reduction

Platform reduction

We can reduce the number of processors used.

Reduction

Platform reduction

We can reduce the number of processors used.

$P_5 : c_5 \geq 5$

$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 8$

Reduction

Platform reduction

We can reduce the number of processors used.

$P_5 : c_5 \geq 5$

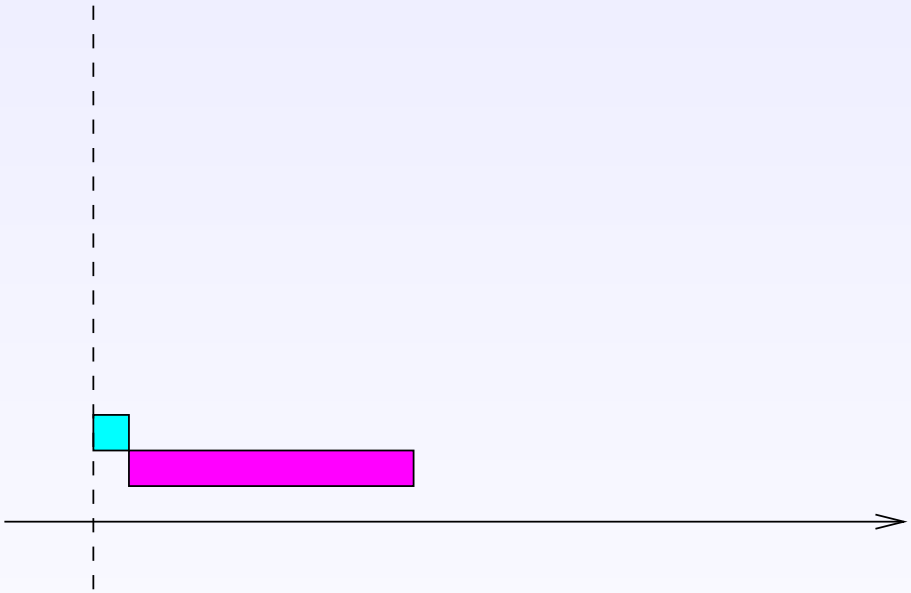
$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 8$



Reduction

Platform reduction

We can reduce the number of processors used.

$P_5 : c_5 \geq 5$

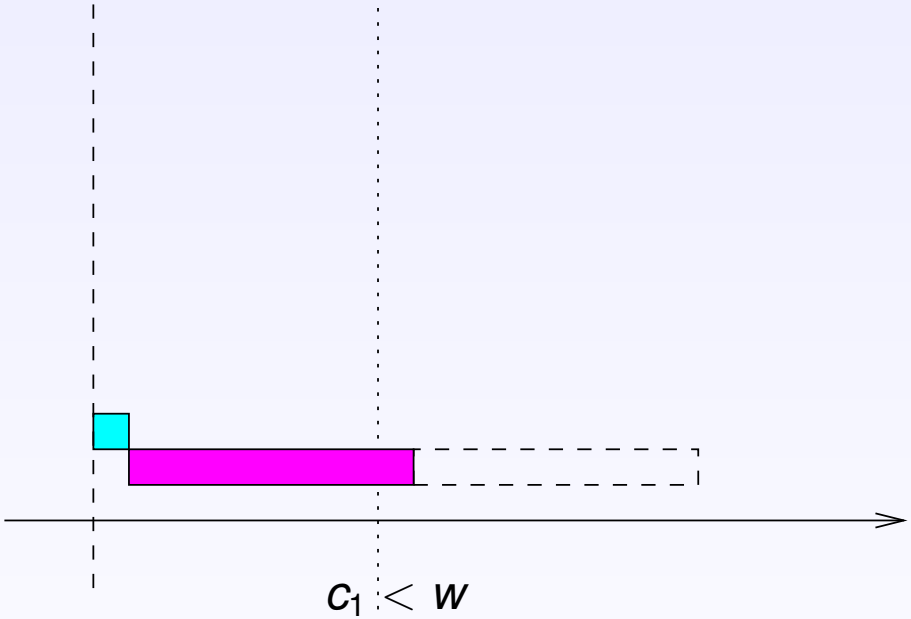
$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 8$



Reduction

Platform reduction

We can reduce the number of processors used.

$P_5 : c_5 \geq 5$

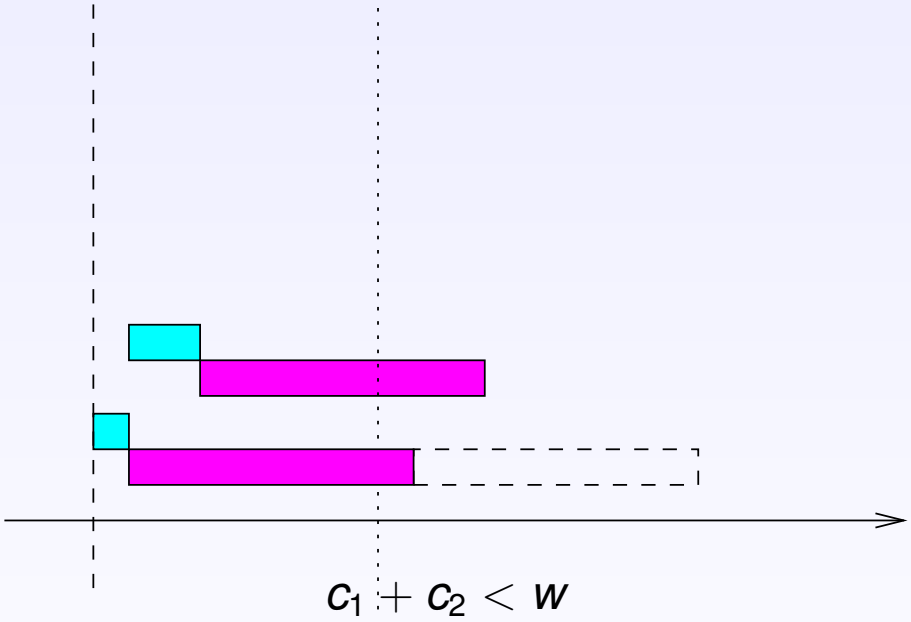
$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 8$



Reduction

Platform reduction

We can reduce the number of processors used.

$P_5 : c_5 \geq 5$

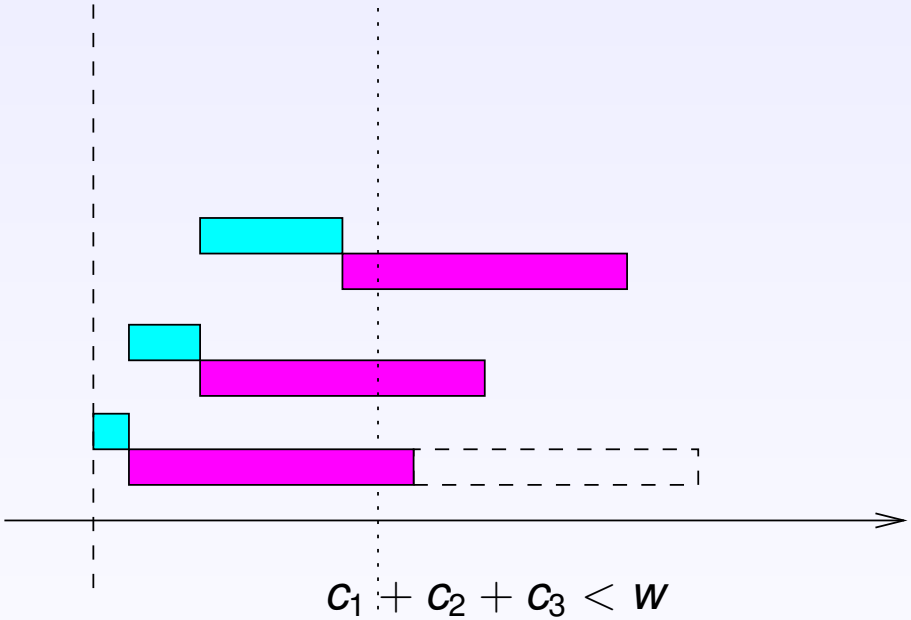
$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 8$



Reduction

Platform reduction

We can reduce the number of processors used.

$P_5 : c_5 \geq 5$

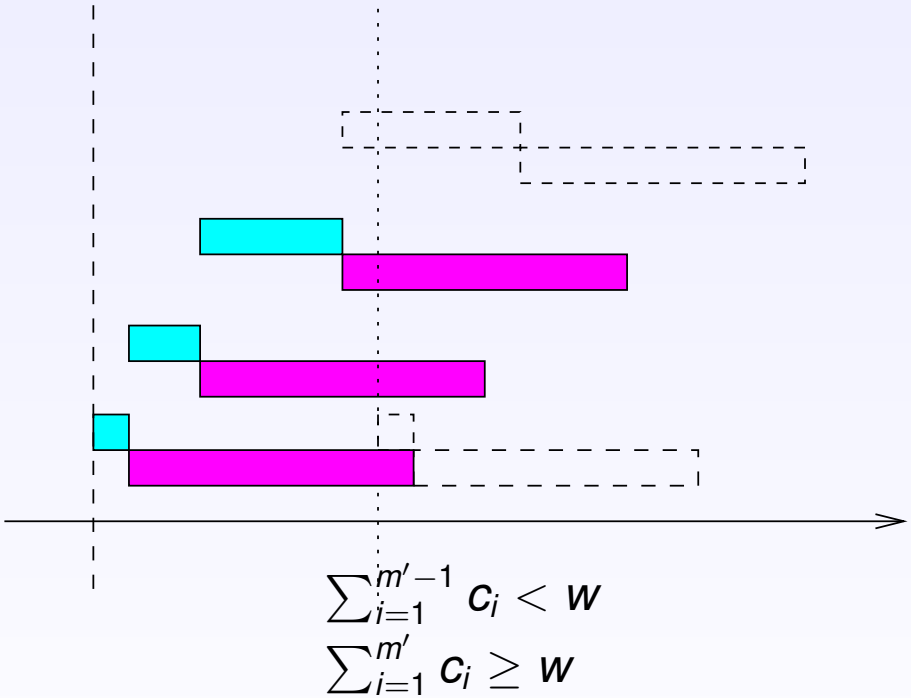
$P_4 : c_4 = 5$

$P_3 : c_3 = 4$

$P_2 : c_2 = 2$

$P_1 : c_1 = 1$

$w = 8$

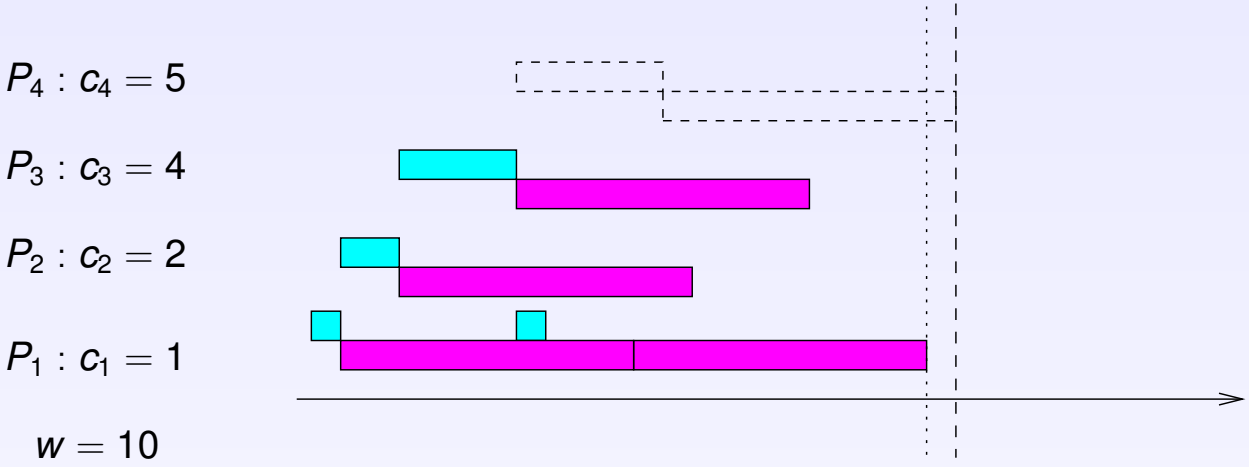


Reduction

But the m' can be useful.

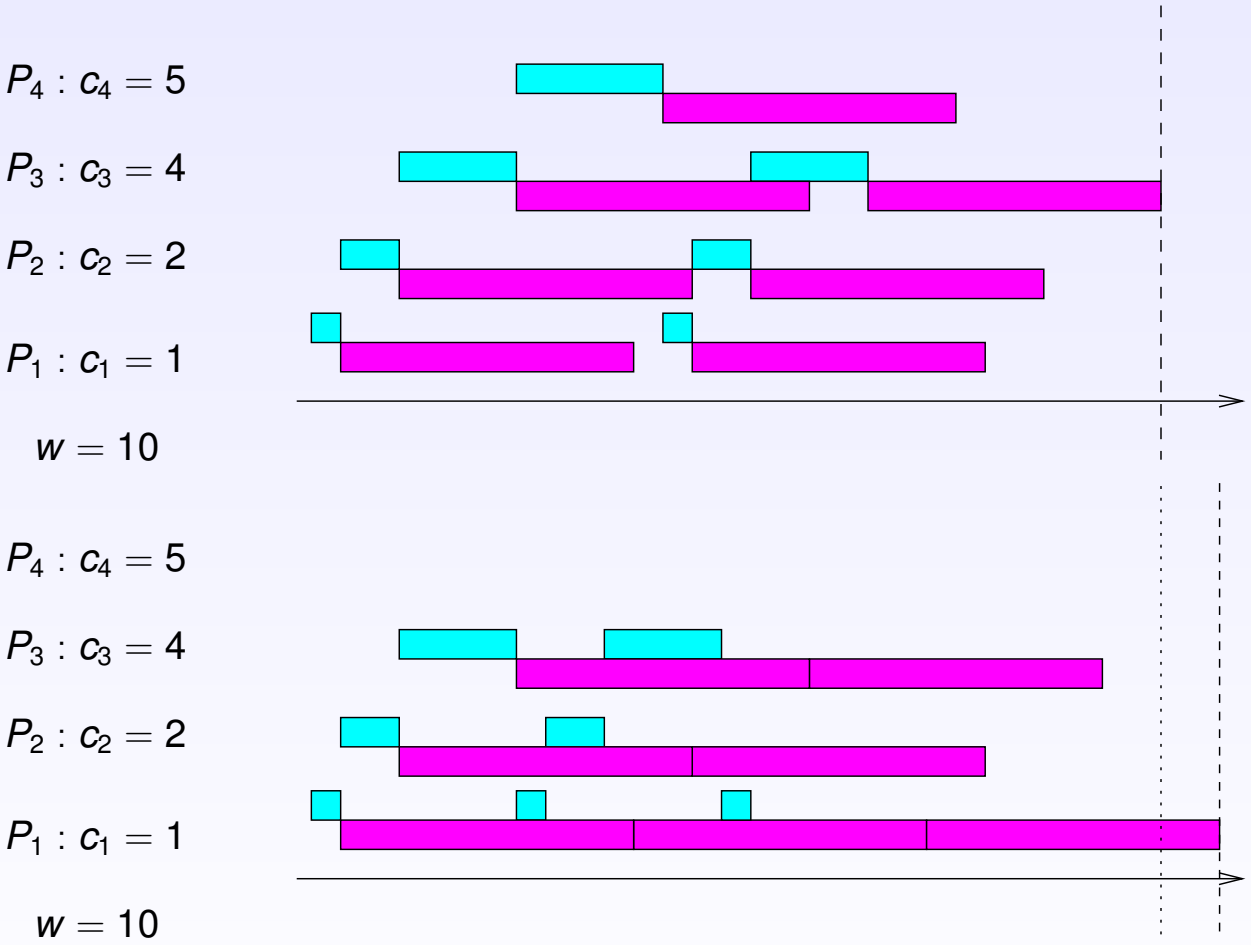
Reduction

But the m' can be useful.



Reduction

But the m' can be useful.



Heuristic

$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$

Heuristic

$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$

Heuristic

$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$

Heuristic

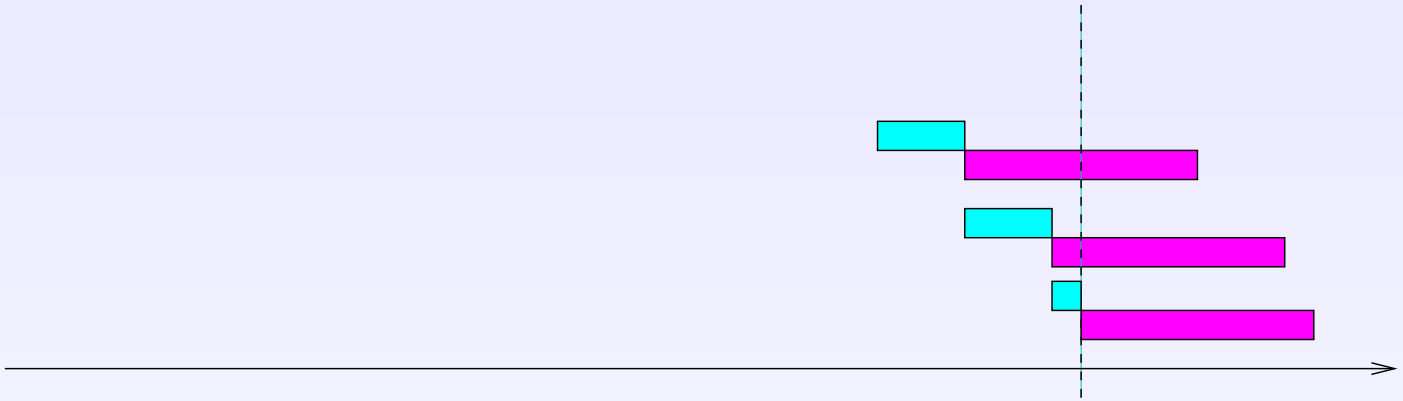
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

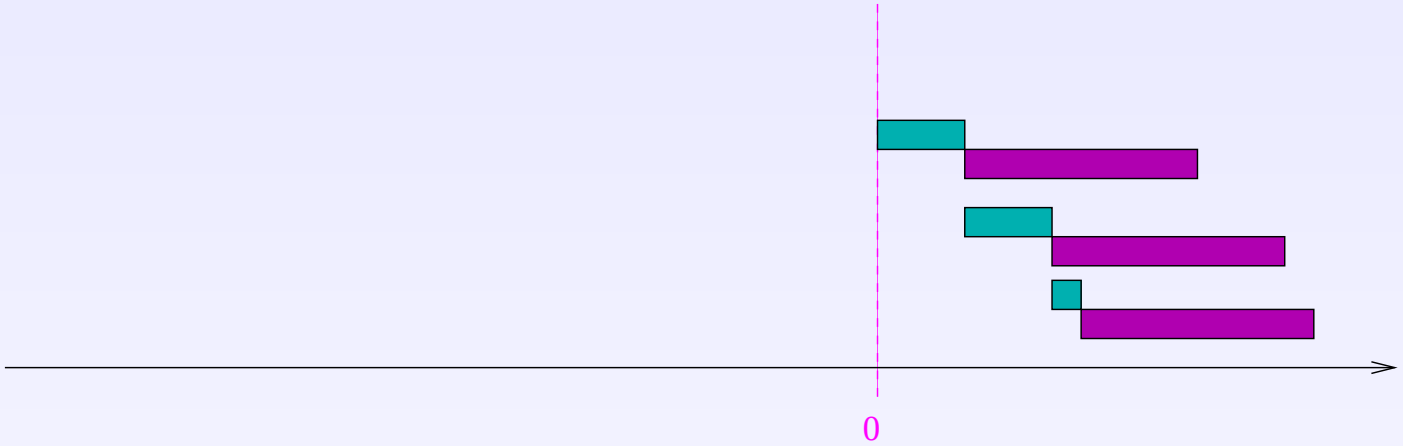
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

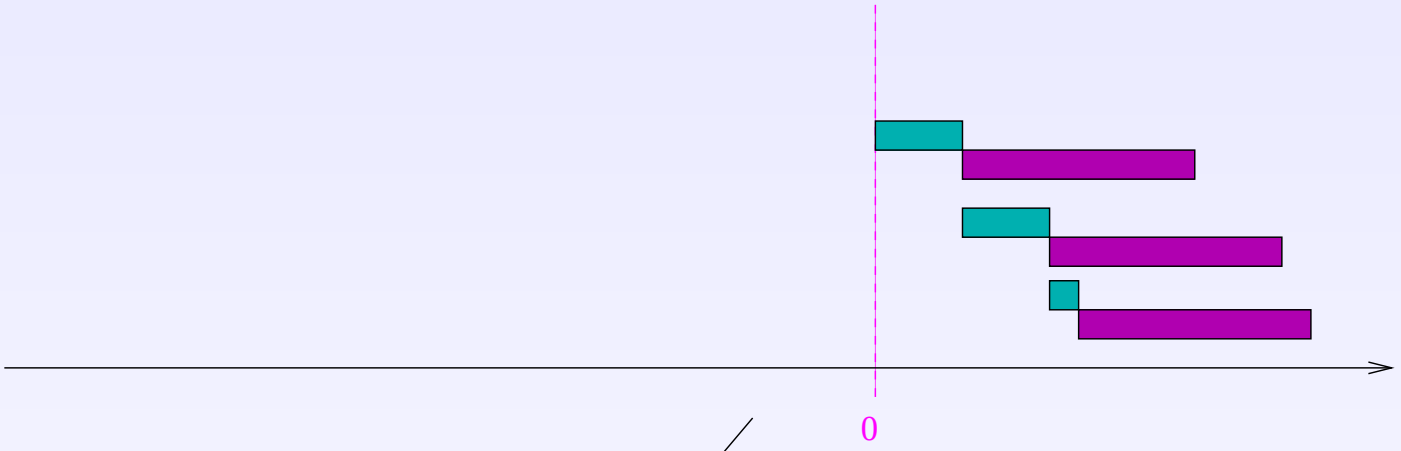
$P_4 : c_4 = 3$

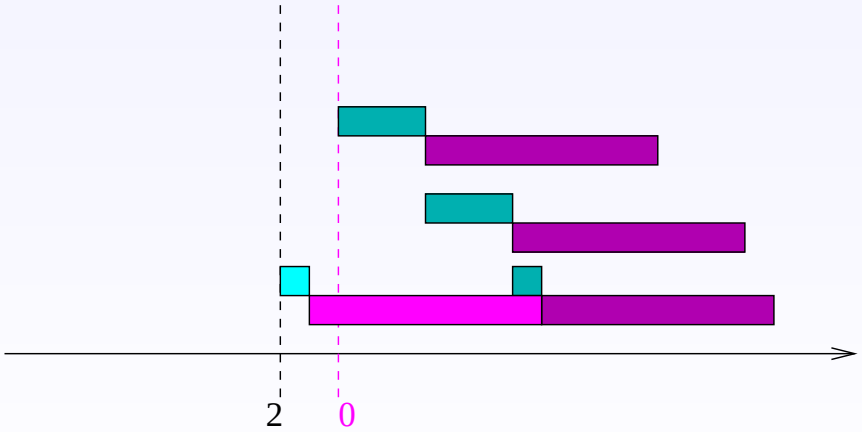
$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$





Heuristic

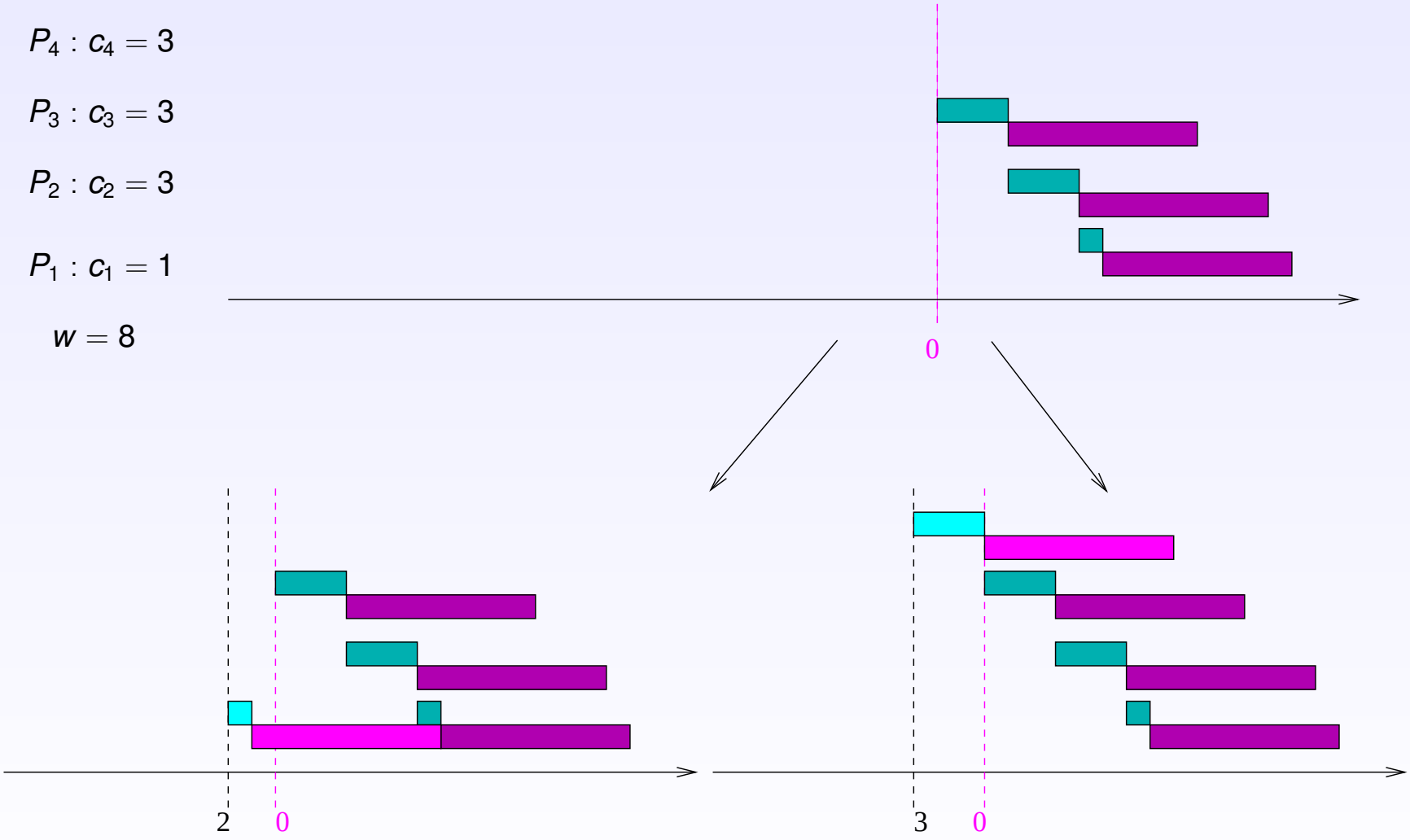
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

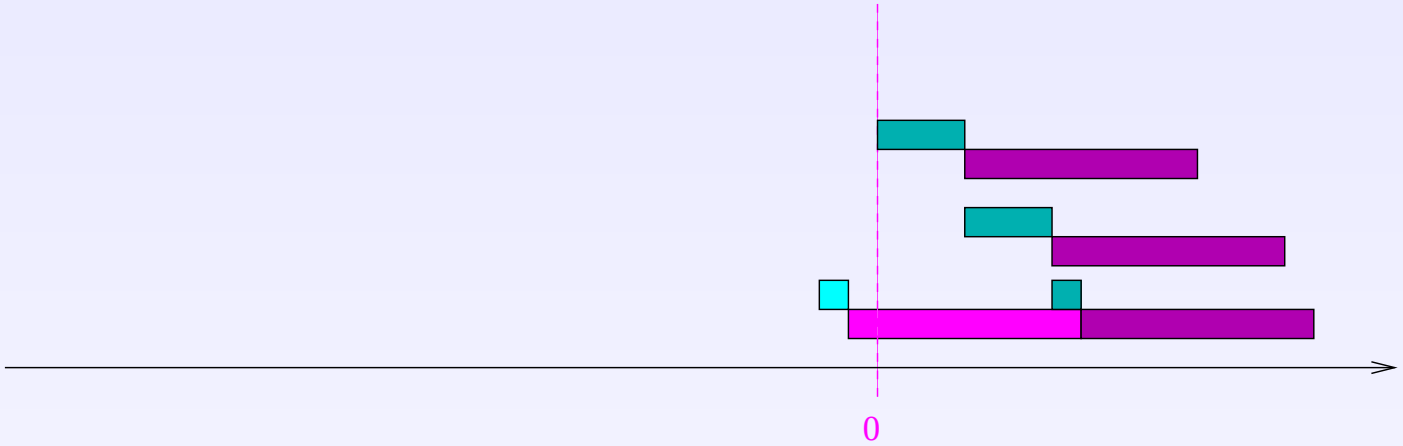
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

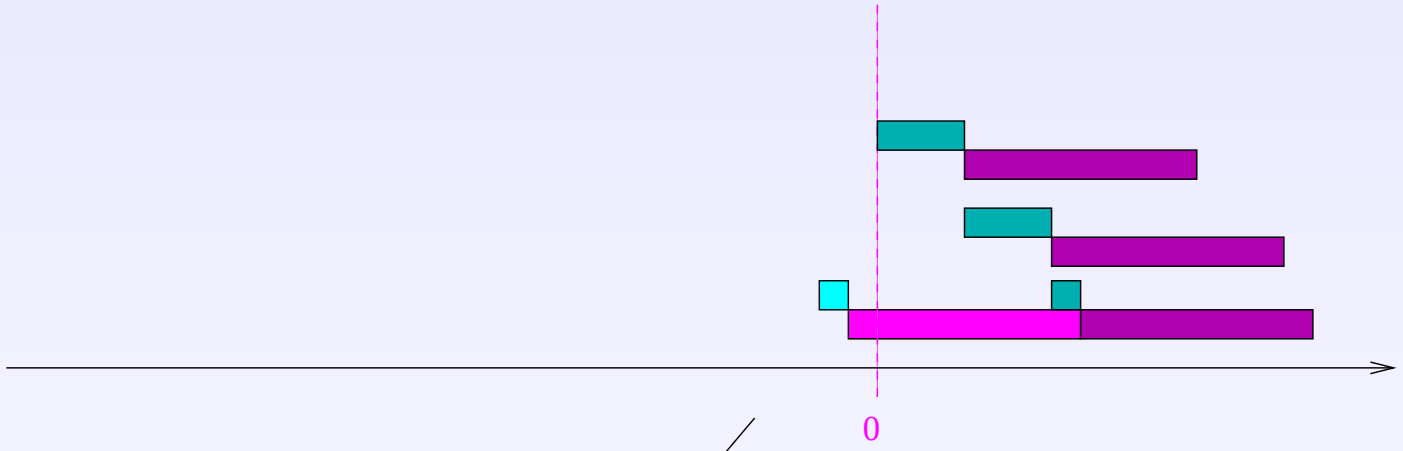
$P_4 : c_4 = 3$

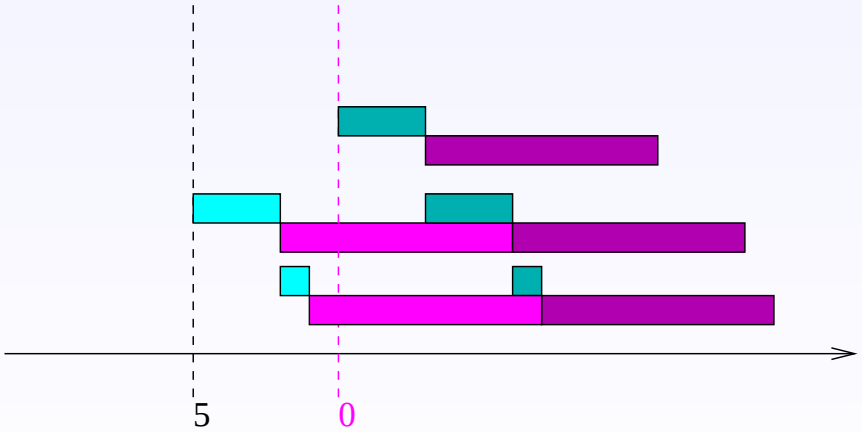
$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$





Heuristic

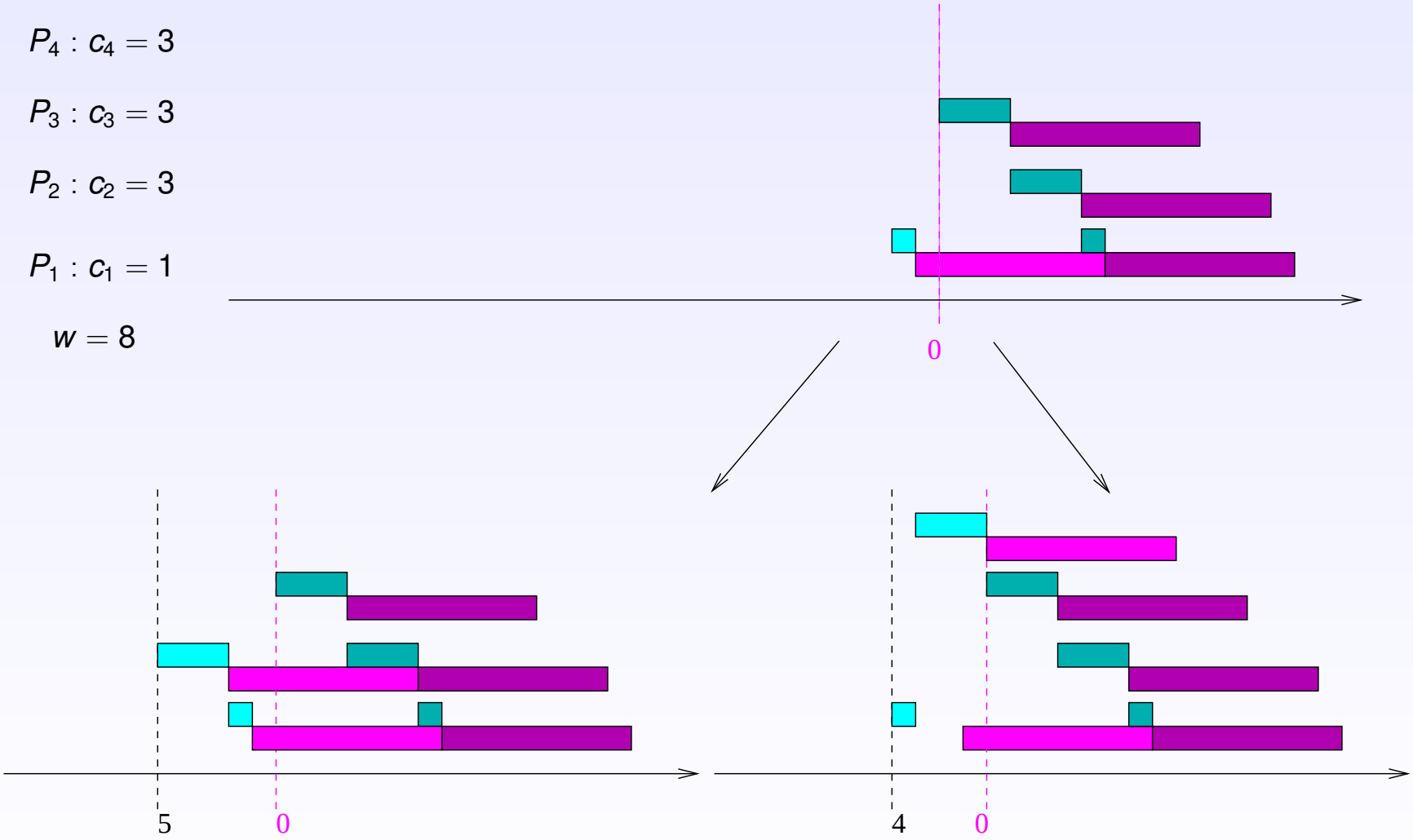
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

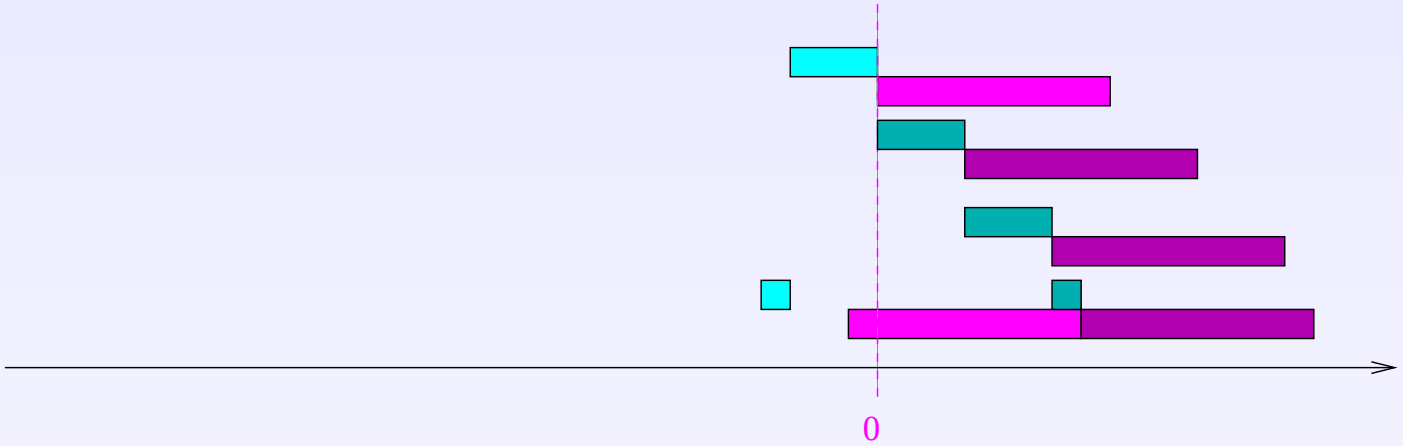
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

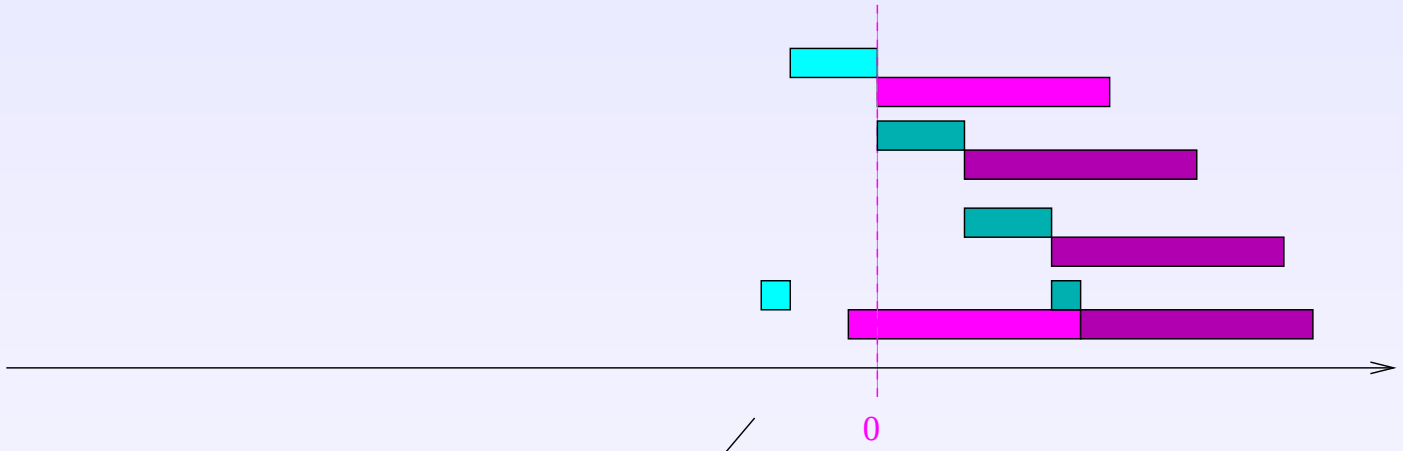
$P_4 : c_4 = 3$

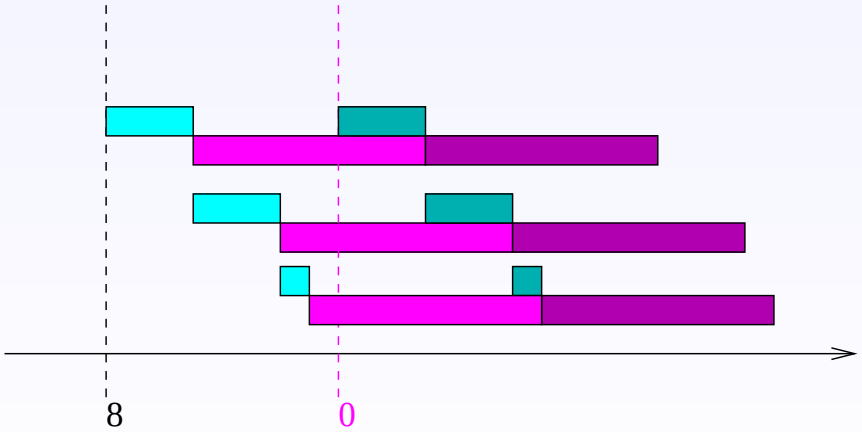
$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

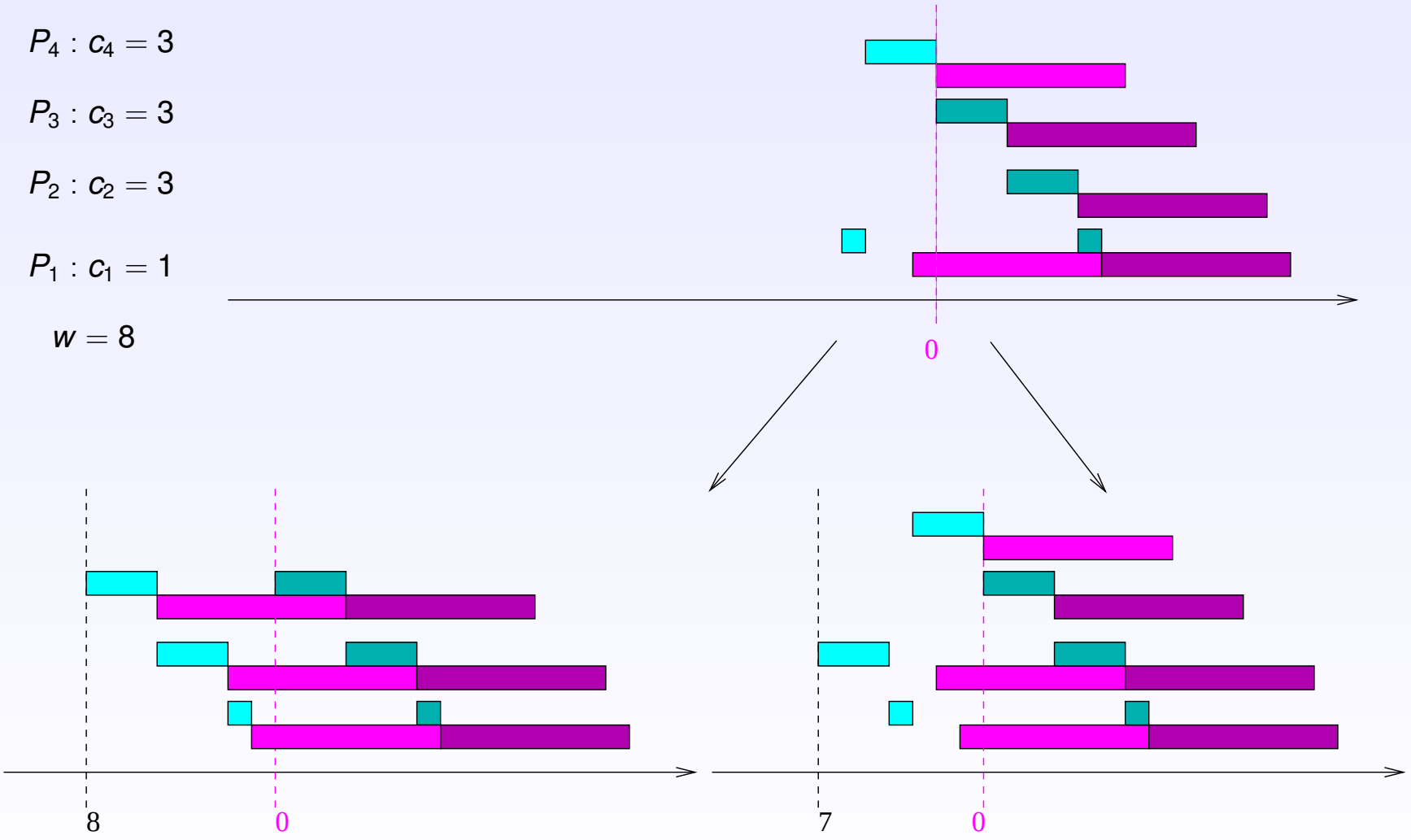
$P_1 : c_1 = 1$

$w = 8$





Heuristic



Heuristic

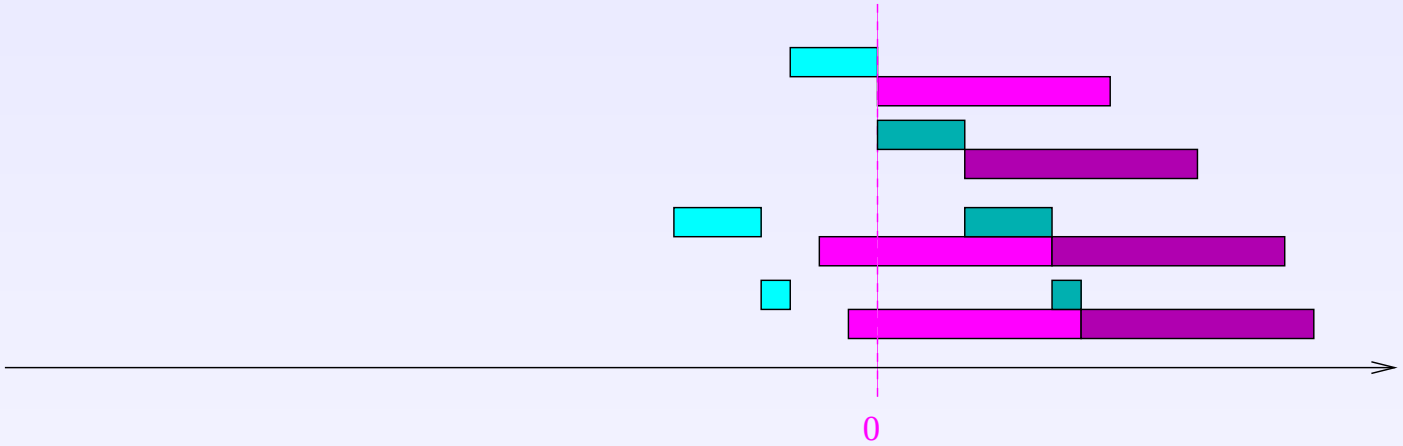
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

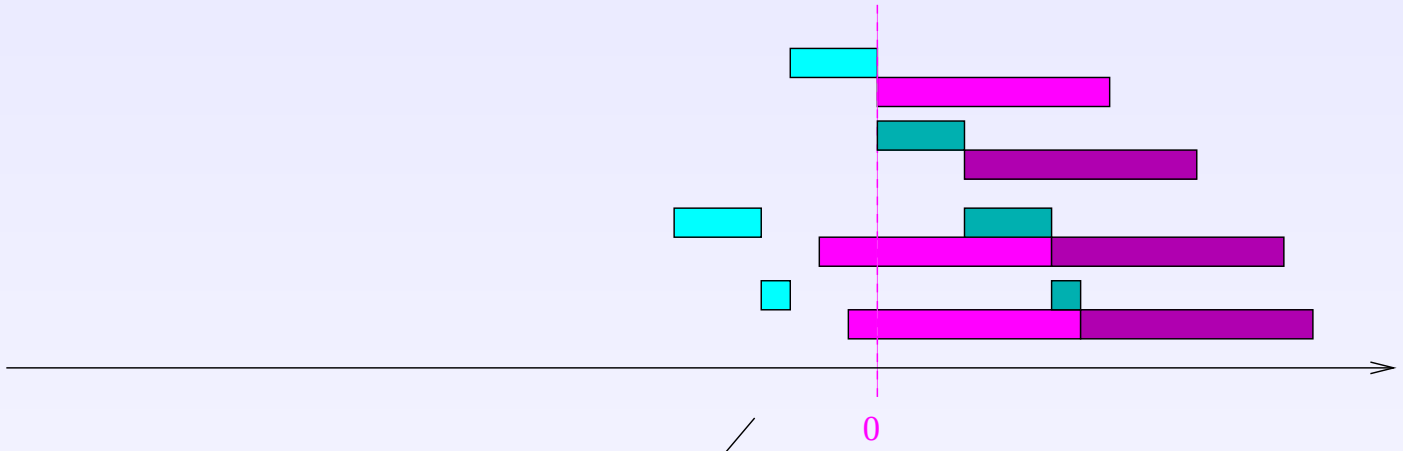
$P_4 : c_4 = 3$

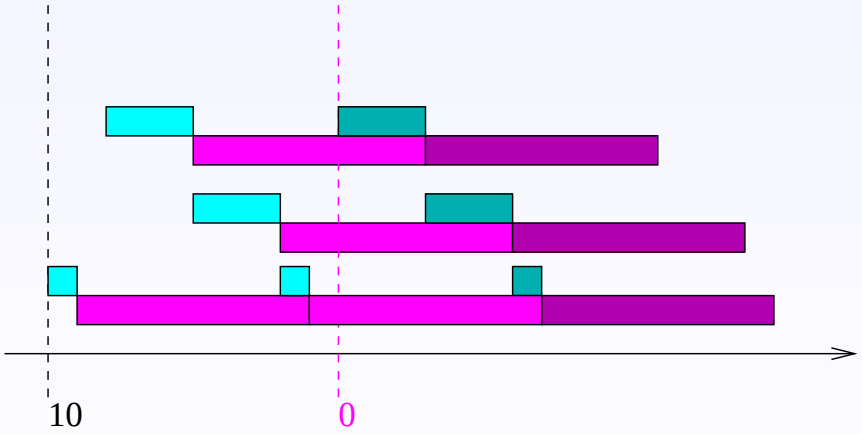
$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

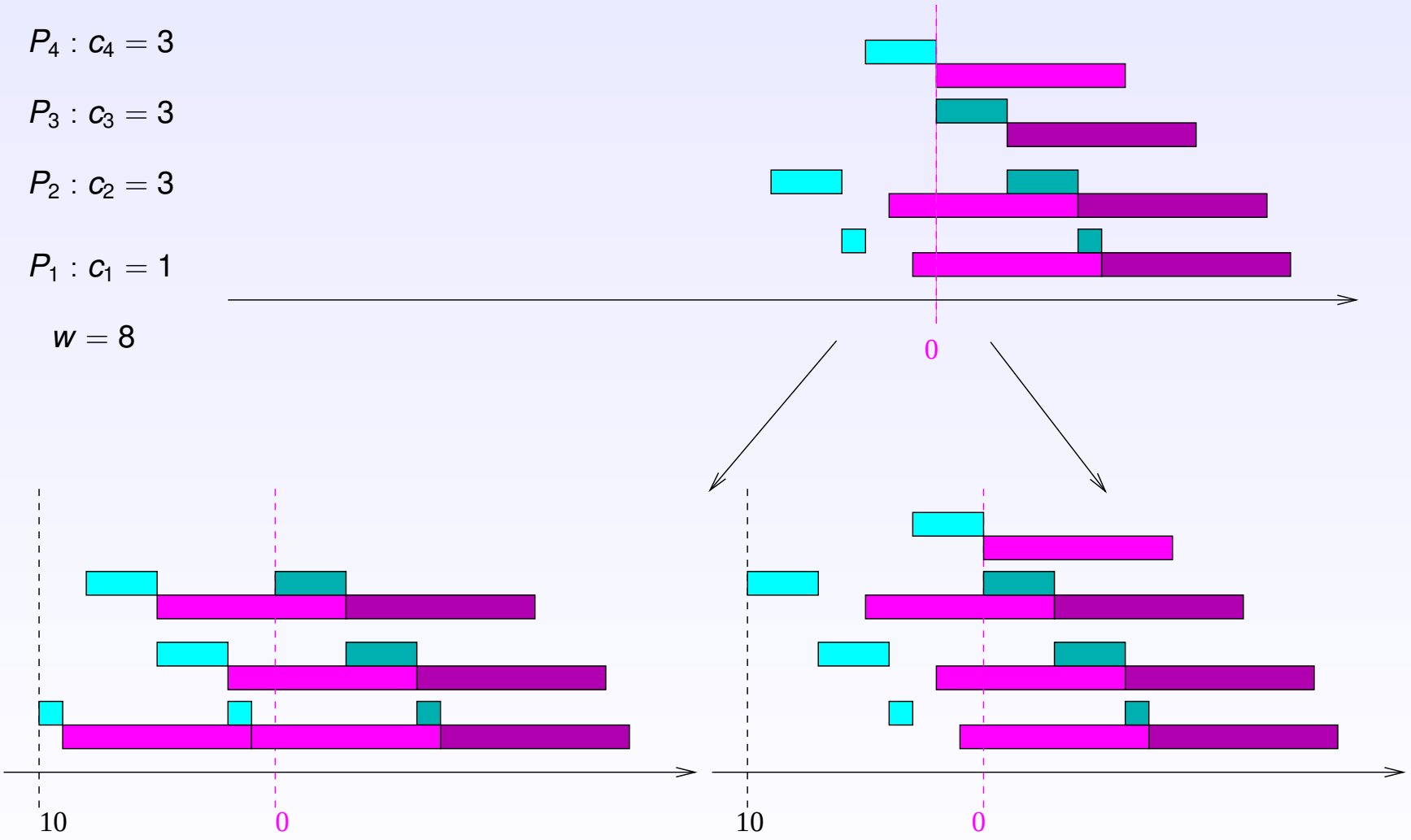
$P_1 : c_1 = 1$

$w = 8$





Heuristic



Heuristic

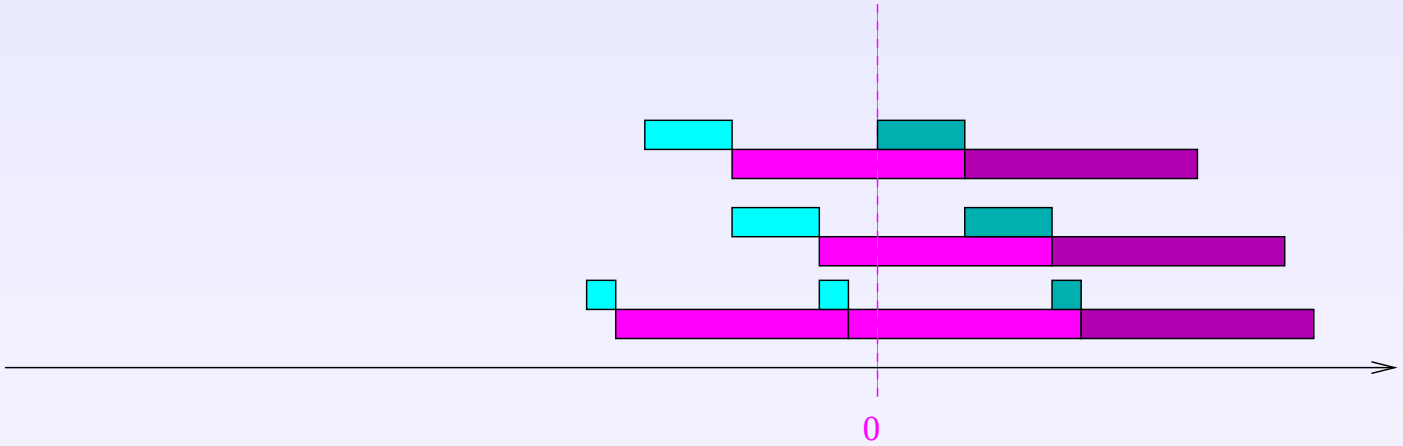
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

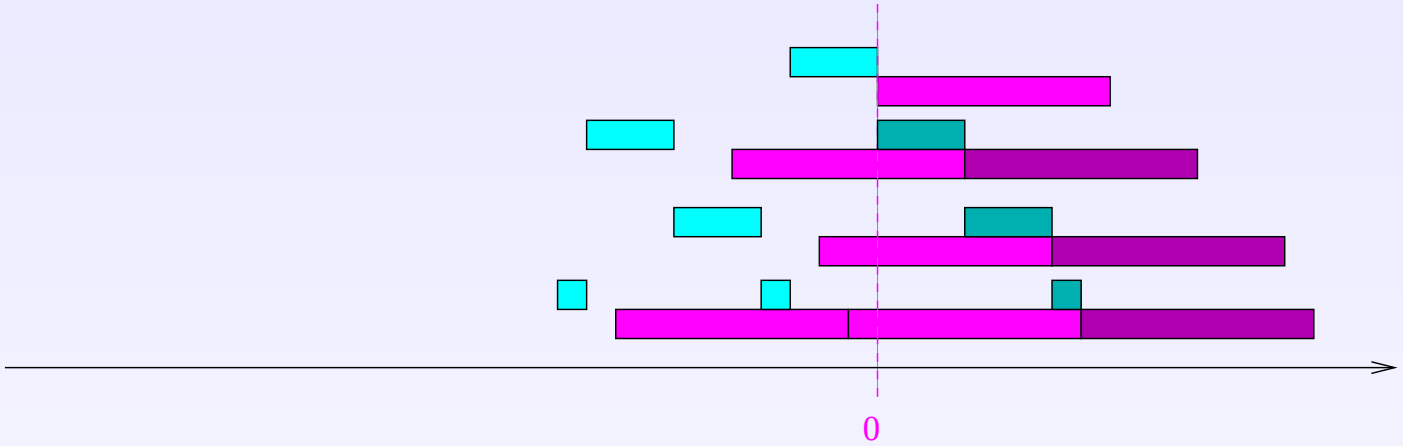
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

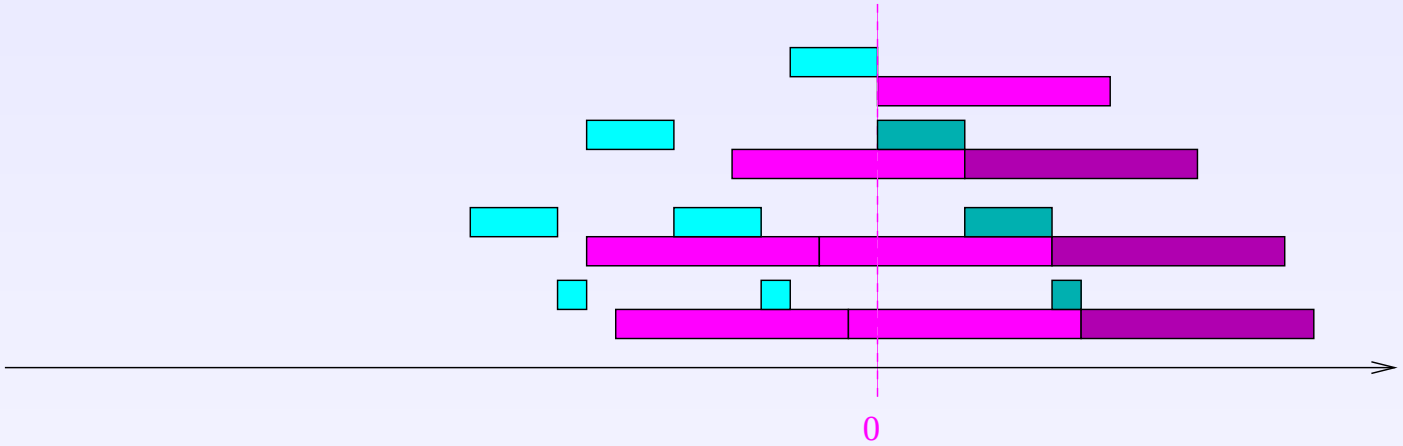
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

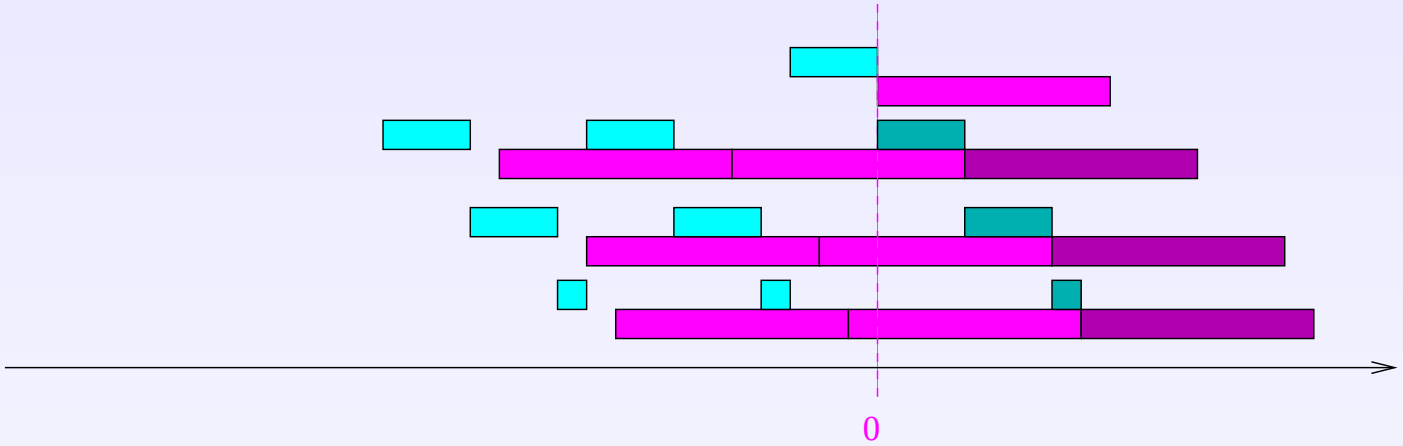
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

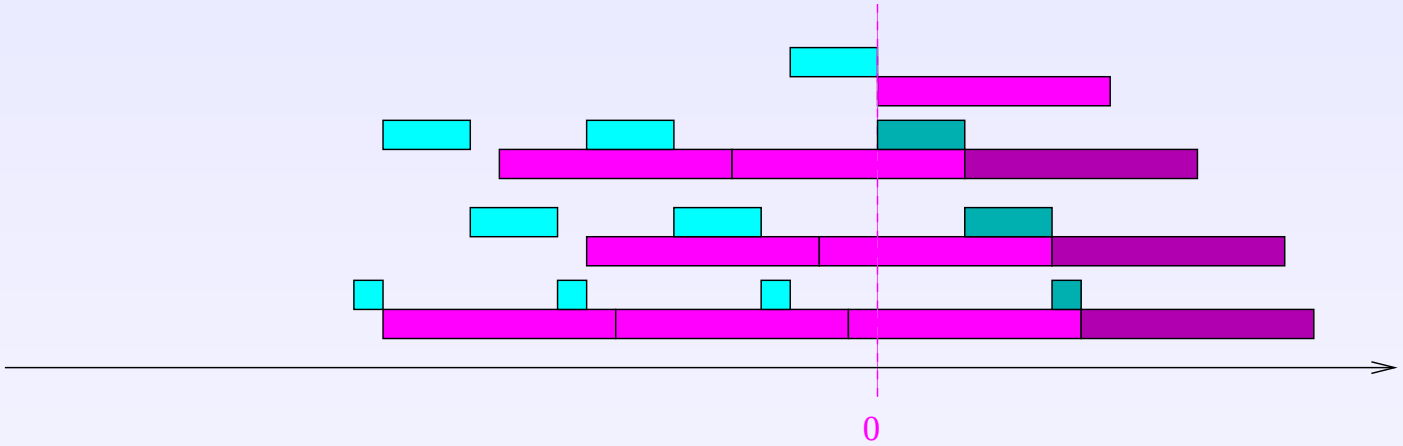
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

$w = 8$



Heuristic

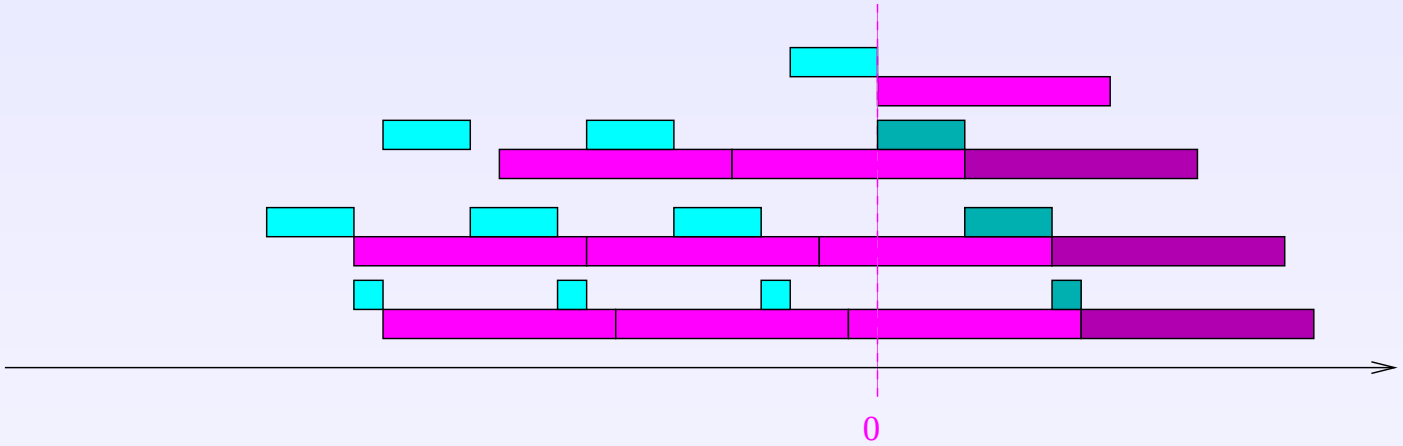
$P_4 : c_4 = 3$

$P_3 : c_3 = 3$

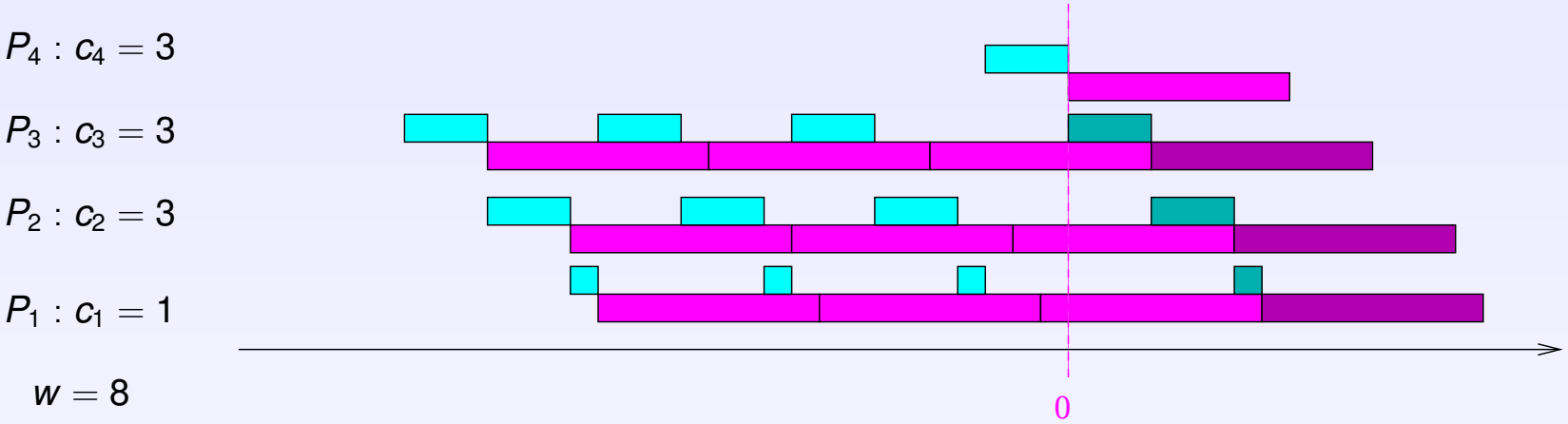
$P_2 : c_2 = 3$

$P_1 : c_1 = 1$

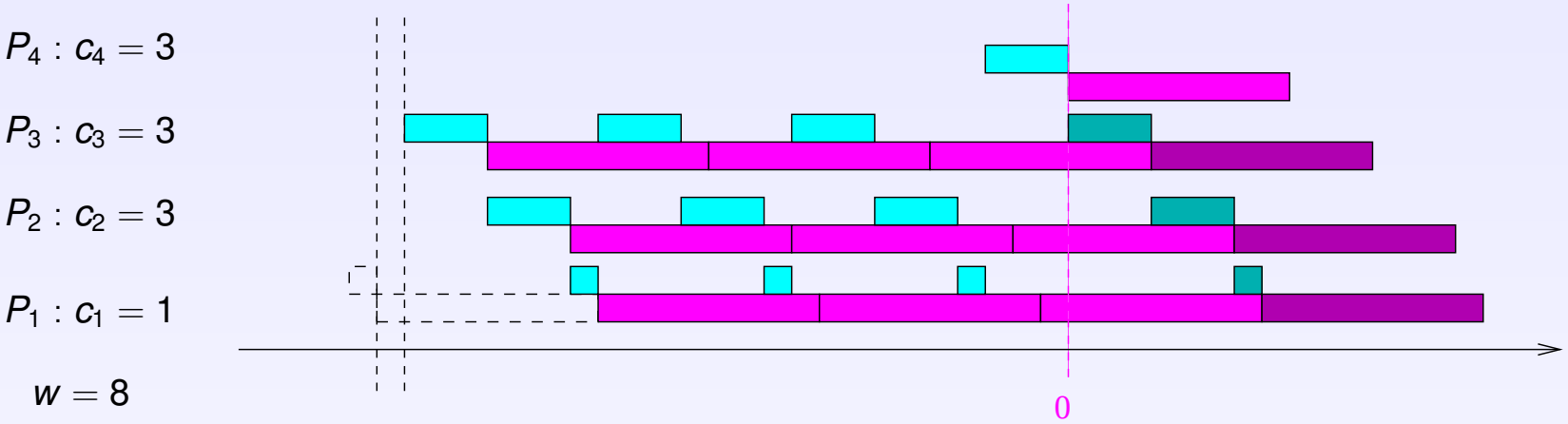
$w = 8$



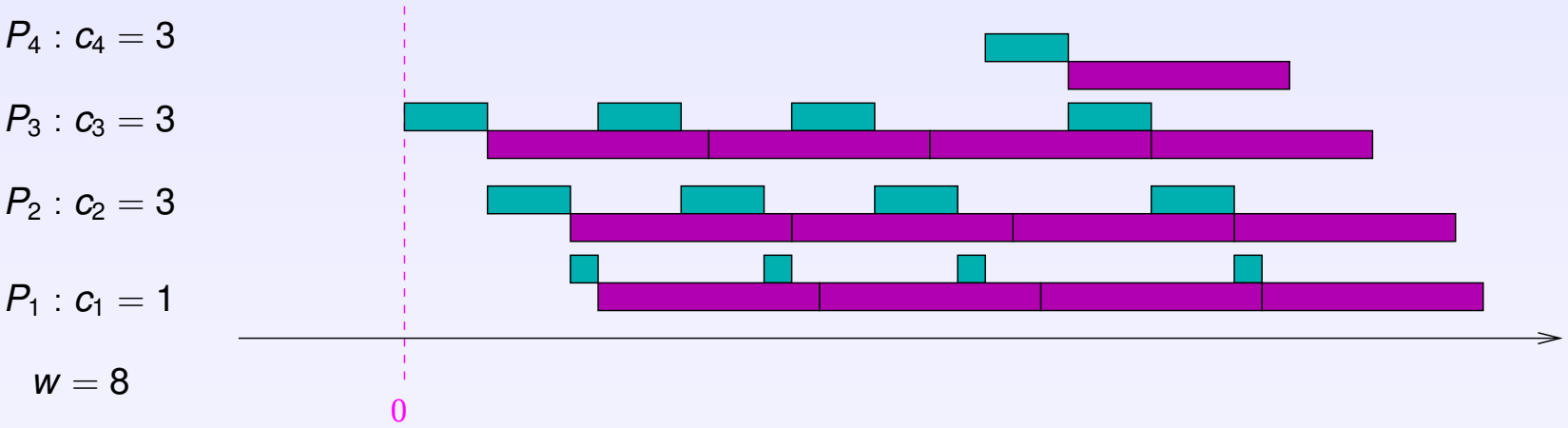
Heuristic



Heuristic



Heuristic



Outline

The platform

Hardware

- 5 computers (1 master, 4 slaves)
- 1 Fast-Ethernet switch

Software

- MPI communications
- Modification of slave parameters

Tasks

Computation of matrices determinant

Algorithms

- Algorithm **1** is *SRPT*
- Algorithms **2** is *List Scheduling*
- Algorithms **3, 4, 5** are variant of *Round Robin*
- Algorithms **6, 7** are respectively built for minimizing makespan on communication homogeneous and computation homogeneous platforms.

Algorithms

Properties

- Algorithm **1** is a dynamic one
- Algorithms **4** and **7** only take into account communication heterogeneity
- Algorithms **5** and **6** only take into account computation heterogeneity
- Algorithms **2** and **3** take into account both communication and computation heterogeneity

Results

Heterogeneous computation:

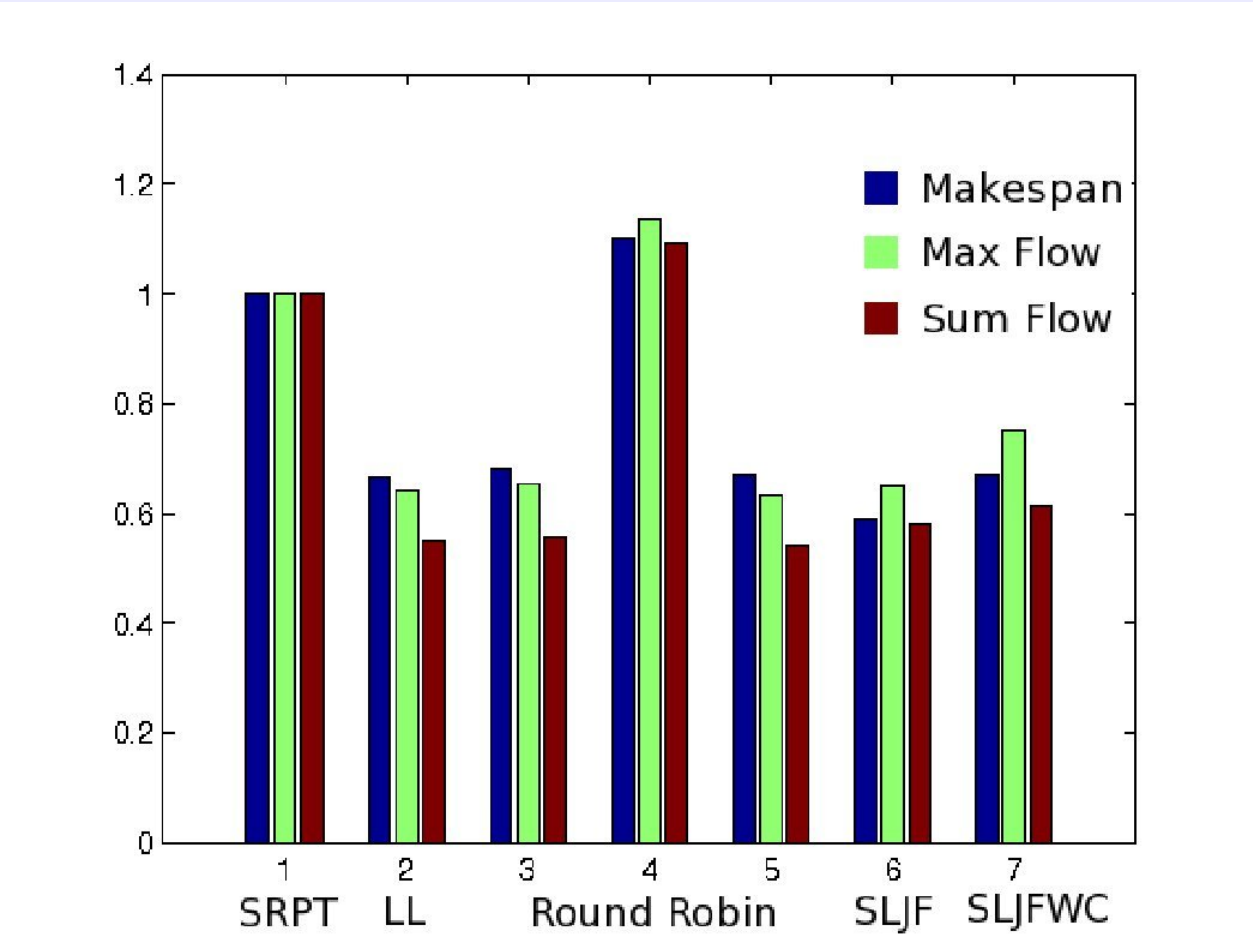


Figure: Normalized objective functions

Results

Heterogeneous communication:

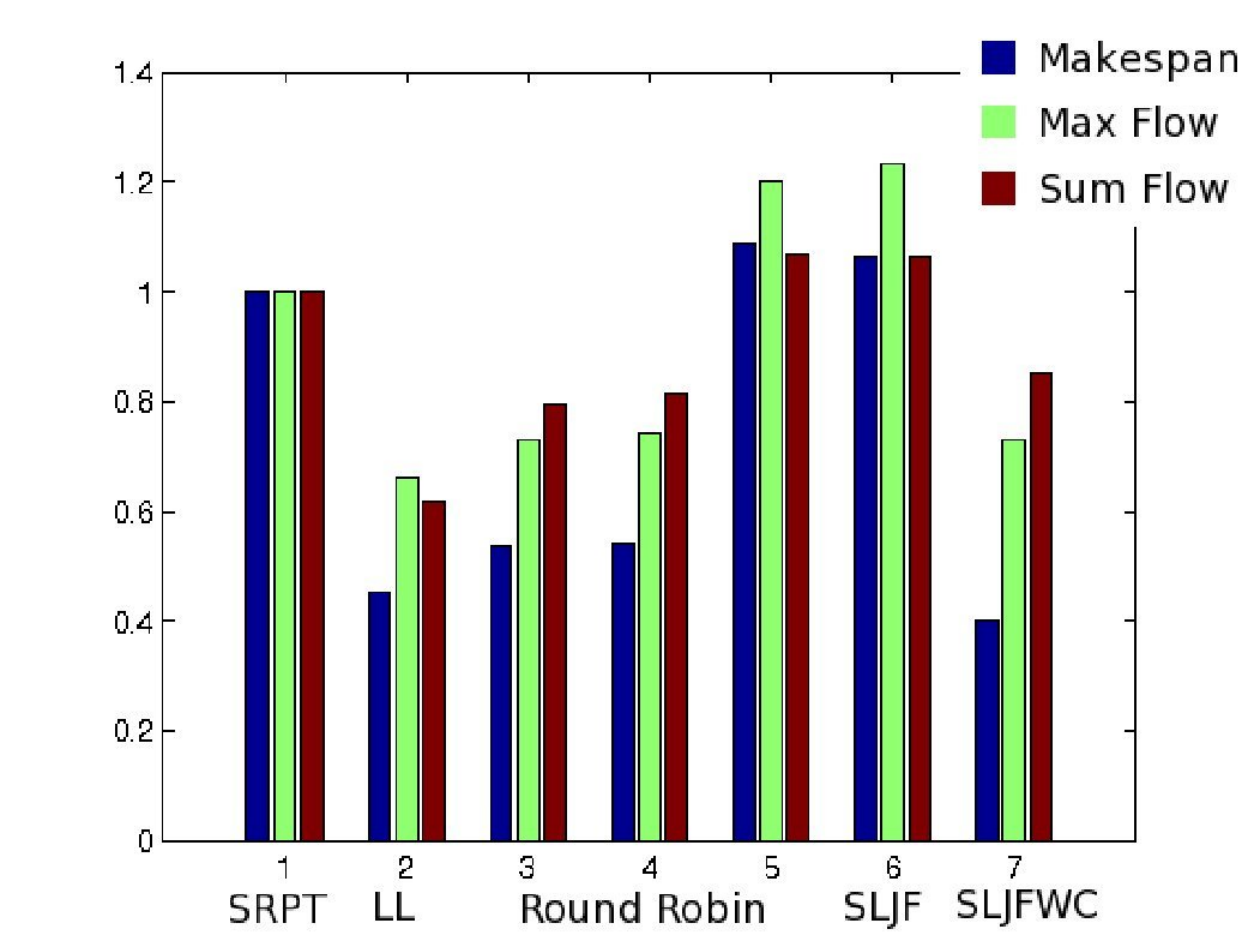


Figure: Normalized objective functions

Results

General case:

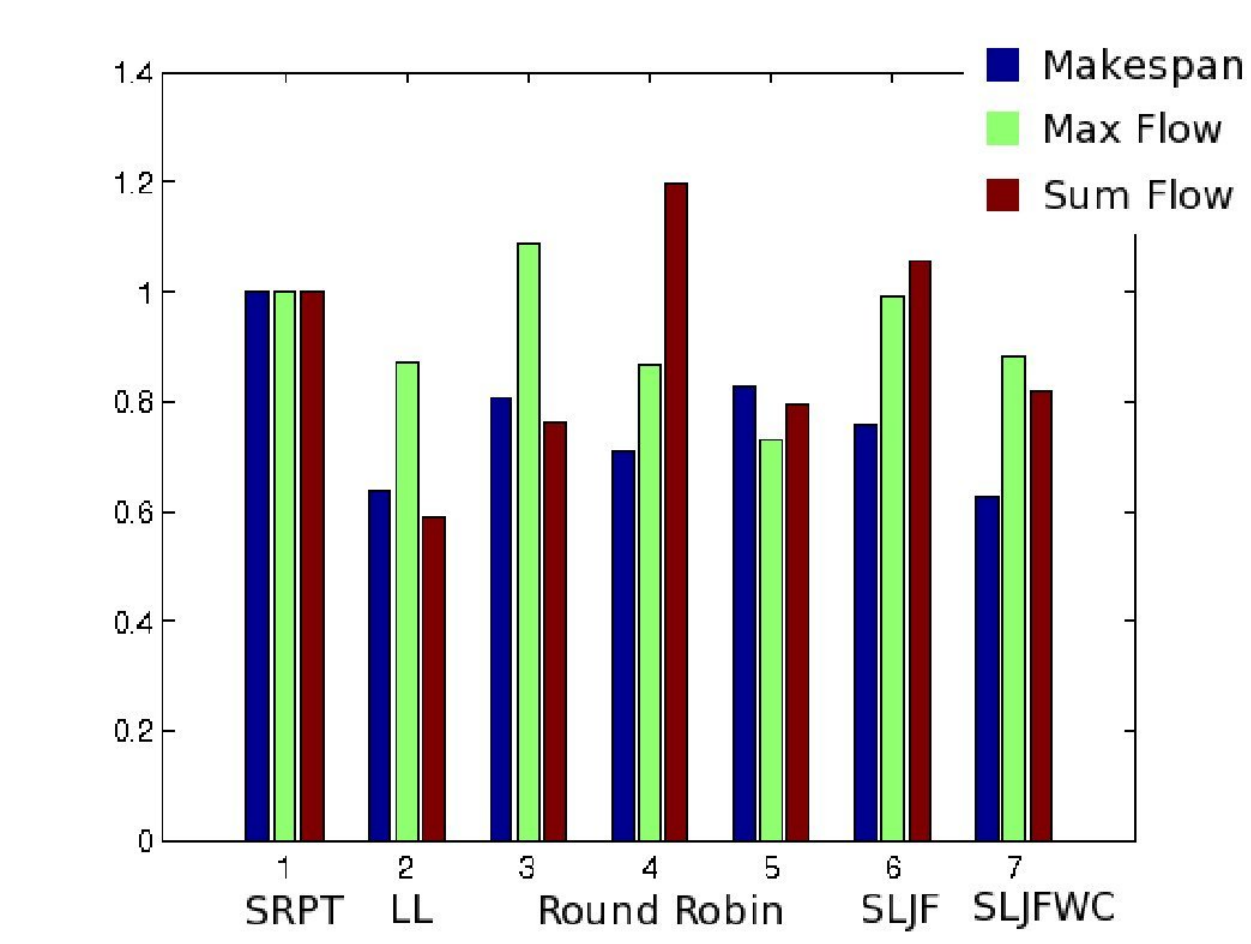


Figure: Normalized objective functions

Results

Summary

- Dynamic algorithm is outperformed by static algorithms on stable platform, because of communication delay
- One heuristic meant to be used on a computation homogeneous platform is better than the other most of the time (95%), and close to the best found algorithm (2%) elsewhere

Point out the importance to take into account the relative speed of communication links when searching a close-to-optimal solution to our scheduling problem.

Results

Summary

- Dynamic algorithm is outperformed by static algorithms on stable platform, because of communication delay
- One heuristic meant to be used on a computation homogeneous platform is better than the other most of the time (95%), and close to the best found algorithm (2%) elsewhere

Point out the importance to take into account the relative speed of communication links when searching a close-to-optimal solution to our scheduling problem.

Outline

Contributions and perspectives

Contributions

- Comprehensive set of lower bounds for the competitive ratio of any scheduling algorithm, for each source of heterogeneity and for each target objective function,
- Experiments on real small-size master-slave platform.

Perspectives

- See which bounds can be met, if any, and design the corresponding approximation algorithms,
- Theoretical study of off-line scheduling problems,
- Detailed comparison of all previous heuristics on larger platforms,
- Widen the scope of the MPI experiments.

Contributions and perspectives

Contributions

- Comprehensive set of lower bounds for the competitive ratio of any scheduling algorithm, for each source of heterogeneity and for each target objective function,
- Experiments on real small-size master-slave platform.

Perspectives

- See which bounds can be met, if any, and design the corresponding approximation algorithms,
- Theoretical study of off-line scheduling problems,
- Detailed comparison of all previous heuristics on larger platforms,
- Widen the scope of the MPI experiments.

Thank you

Any question?