Arnaud LEGRAND

École Normale Supérieure de Lyon,

Laboratoire de l'Informatique du Parallélisme

# The Master-Slave Paradigm with Heterogeneous Processors

Olivier BEAUMONT, Arnaud LEGRAND, and Yves ROBERT

# Contents

# Contents

# Contents

# Contents

# Master-slave tasking

**Master-slave tasking** Simple yet widely used technique.

**Standard implementation** Independent tasks executed by identical processors (the slave) under the centralized supervision of a control processor (the master).

**Heterogeneous implementation** Slave processors have different computation speeds.

**Applications** Any Monte Carlo simulation:

- $\leadsto$ cellular microphysiology [SBSS98],
- $\leadsto$ reactor simulations [WT98],
- $\leadsto$ protein conformations [SFM$^+$]...

# Master-slave tasking

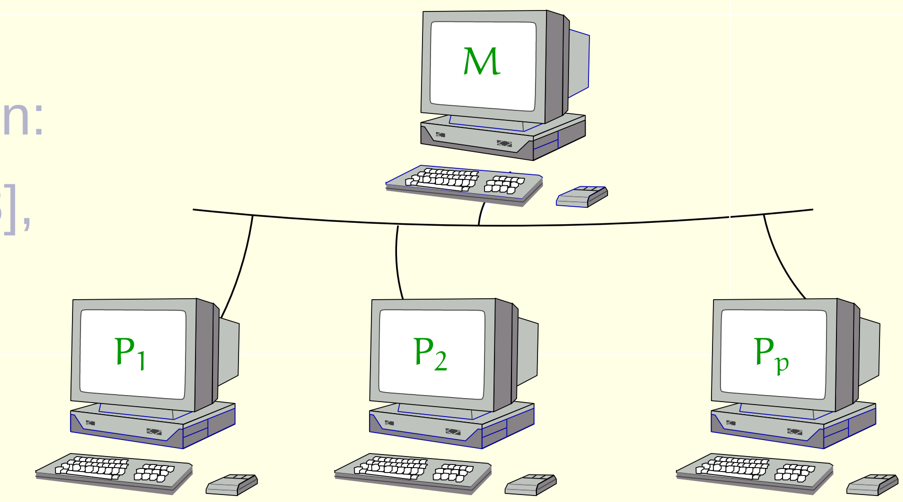
**Master-slave tasking** Simple yet widely used technique.

**Standard implementation** Independent tasks executed by identical processors (the slave) under the centralized supervision of a control processor (the master).

**Heterogeneous implementation** Slave processors have different computation speeds.

**Applications** Any Monte Carlo simulation:

- ⇝ cellular microphysiology [SBSS98],
- ⇝ reactor simulations [WT98],
- ⇝ protein conformations [SFM+]...



# Master-slave tasking

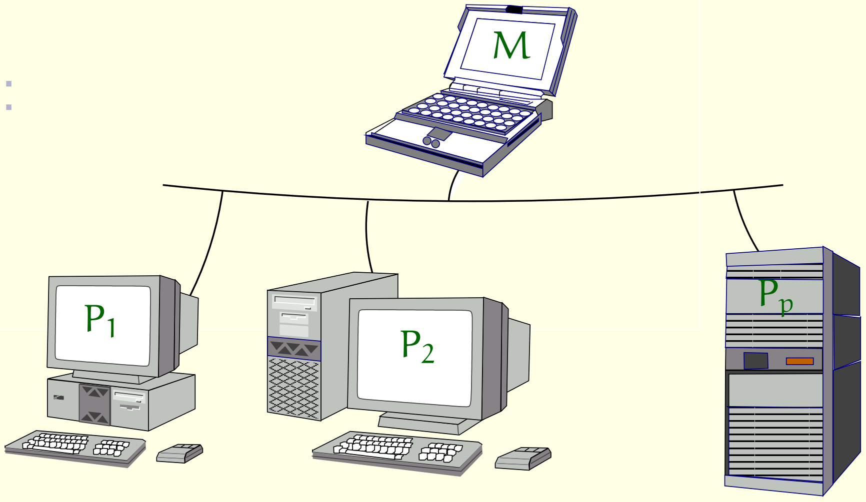
**Master-slave tasking** Simple yet widely used technique.

**Standard implementation** Independent tasks executed by identical processors (the slave) under the centralized supervision of a control processor (the master).

**Heterogeneous implementation** Slave processors have different computation speeds.

**Applications** Any Monte Carlo simulation:

- $\rightsquigarrow$ cellular microphysiology [SBSS98],
- $\rightsquigarrow$ reactor simulations [WT98],
- $\rightsquigarrow$ protein conformations [SFM$^+$]...



# Master-slave tasking

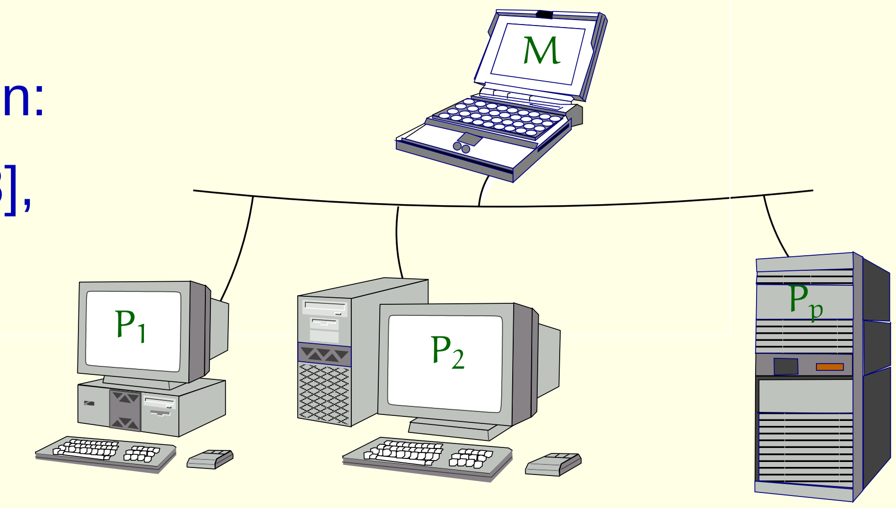
**Master-slave tasking** Simple yet widely used technique.

**Standard implementation** Independent tasks executed by identical processors (the slave) under the centralized supervision of a control processor (the master).

**Heterogeneous implementation** Slave processors have different computation speeds.

**Applications** Any Monte Carlo simulation:

- ⇝ cellular microphysiology [SBSS98],
- ⇝ reactor simulations [WT98],
- ⇝ protein conformations [SFM+]...



# Problem statement

- Pool of independent tasks to be processed by the $p$ slaves.
- All tasks are of same-size, i.e. they represent the same amount of processing.
- Tasks are considered to be atomic (execution cannot be preempted once initiated).
- Communication through a shared medium, that can be accessed only in exclusive mode.

# Problem statement

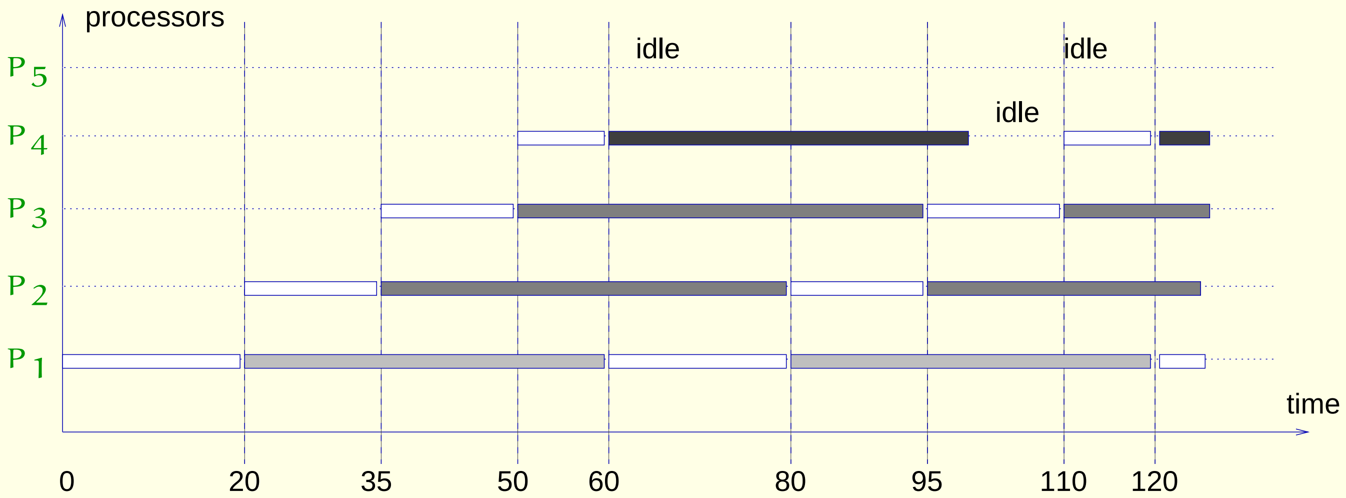
- Pool of independent tasks to be processed by the $p$ slaves.
- All tasks are of same-size, i.e. they represent the same amount of processing.
- Tasks are considered to be atomic (execution cannot be preempted once initiated).
- Communication through a shared medium, that can be accessed only in exclusive mode.

# Problem statement

- Pool of independent tasks to be processed by the $p$ slaves.
- All tasks are of same-size, i.e. they represent the same amount of processing.
- Tasks are considered to be atomic (execution cannot be preempted once initiated).
- Communication through a shared medium, that can be accessed only in exclusive mode.

# Problem statement

- Pool of independent tasks to be processed by the $p$ slaves.
- All tasks are of same-size, i.e. they represent the same amount of processing.
- Tasks are considered to be atomic (execution cannot be preempted once initiated).
- Communication through a shared medium, that can be accessed only in exclusive mode.



# Heterogeneous platform : Optimization problems

Processors are heterogeneous: slave $P_i$ requires $t_i$ units of time to process a single task. Each $P_i$ will execute $c_i$ tasks (where $c_i$ is to be determined) from the pool.

**MinTime($C$)** Given a total number of tasks $C$, determine the best allocation of tasks to slaves, i.e. the allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. $\sum_{i=1}^{p} c_i = C$ and which minimizes the total execution time.

**MaxTasks($T$)** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. the allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and $\sum_{i=1}^{p} c_i = C$ is maximized.

We concentrate on solving the second problem **MaxTasks($T$)**.

# Heterogeneous platform : Optimization problems

Processors are heterogeneous: slave $P_i$ requires $t_i$ units of time to process a single task. Each $P_i$ will execute $c_i$ tasks (where $c_i$ is to be determined) from the pool.

**MinTime($C$)** Given a total number of tasks $C$, determine the best allocation of tasks to slaves, i.e. the allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. $\sum_{i=1}^{p} c_i = C$ and which minimizes the total execution time.

**MaxTasks($T$)** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. the allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and $\sum_{i=1}^{p} c_i = C$ is maximized.

We concentrate on solving the second problem **MaxTasks($T$)**.

# Heterogeneous platform : Optimization problems

Processors are heterogeneous: slave $P_i$ requires $t_i$ units of time to process a single task. Each $P_i$ will execute $c_i$ tasks (where $c_i$ is to be determined) from the pool.

**MinTime($C$)** Given a total number of tasks $C$, determine the best allocation of tasks to slaves, i.e. the allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. $\sum_{i=1}^{p} c_i = C$ and which minimizes the total execution time.

**MaxTasks($T$)** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. the allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and $\sum_{i=1}^{p} c_i = C$ is maximized.

We concentrate on solving the second problem **MaxTasks($T$)**.

# Heterogeneous platform : Optimization problems

Processors are heterogeneous: slave $P_i$ requires $t_i$ units of time to process a single task. Each $P_i$ will execute $c_i$ tasks (where $c_i$ is to be determined) from the pool.

**MinTime($C$)** Given a total number of tasks $C$, determine the best allocation of tasks to slaves, i.e. the allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. $\sum_{i=1}^{p} c_i = C$ and which minimizes the total execution time.

**MaxTasks($T$)** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. the allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and $\sum_{i=1}^{p} c_i = C$ is maximized.

We concentrate on solving the second problem **MaxTasks($T$)**.

# Without any communication cost

Assume first that there is no communication cost at all. It is not difficult to solve both previous problems using a greedy algorithm.

The solution of problem **MaxTasks($T$)** is straightforward: we let $c_i = \left\lfloor \frac{T}{t_i} \right\rfloor$ for all $i$, $1 \leq i \leq p$, which obviously defines the optimal solution.

With an initial scattering of data

# Problem statement

Formulation of this problem is taken from Andonie et al. [ACGG98] (implementation of distributed backpropagation neural networks on heterogeneous networks of workstations).

- each slave must receive some data file from the master processor
- each slave must receive the same amount of data $\rightsquigarrow$ each reception costs $t_{\text{com}}$ units of time
- slave $P_i$ process one task in $t_i$ units of time

# Problem statement

Formulation of this problem is taken from Andonie et al. [ACGG98] (implementation of distributed backpropagation neural networks on heterogeneous networks of workstations).

- each slave must receive some data file from the master processor
- each slave must receive the same amount of data $\rightsquigarrow$ each reception costs $t_{\text{com}}$ units of time
- slave $P_i$ process one task in $t_i$ units of time

# Problem statement

Formulation of this problem is taken from Andonie et al. [ACGG98] (implementation of distributed backpropagation neural networks on heterogeneous networks of workstations).

- each slave must receive some data file from the master processor
- each slave must receive the same amount of data $\rightsquigarrow$ each reception costs $t_{\text{com}}$ units of time
- slave $P_i$ process one task in $t_i$ units of time

# Problem statement

Formulation of this problem is taken from Andonie et al. [ACGG98] (implementation of distributed backpropagation neural networks on heterogeneous networks of workstations).

- each slave must receive some data file from the master processor
- each slave must receive the same amount of data $\rightsquigarrow$ each reception costs $t_{\mathrm{com}}$ units of time
- slave $P_i$ process one task in $t_i$ units of time

# A new optimization problem

What are we looking for?

- A permutation $\sigma$ which determines the ordering of the messages from the host: the host sends data to slave $P_i$ at time $\sigma(i)t_{\text{com}}$
- The number of tasks each slave is going to process $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$

**MaxTasks1($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. a permutation $\sigma$ and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma,\ \forall i \in [1,p] : \boxed{\sigma(i)t_{\text{com}} + c_i t_i \leq T} \right)$$

# A new optimization problem

What are we looking for?

- A permutation $\sigma$ which determines the ordering of the messages from the host: the host sends data to slave $P_i$ at time $\sigma(i)t_{\text{com}}$
- The number of tasks each slave is going to process $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$

**MaxTasks1($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. a permutation $\sigma$ and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma,\ \forall i \in [1,p] : \boxed{\sigma(i)t_{\text{com}} + c_i t_i \leq T} \right)$$

# A new optimization problem

What are we looking for?

- A permutation $\sigma$ which determines the ordering of the messages from the host: the host sends data to slave $P_i$ at time $\sigma(i)t_{\text{com}}$
- The number of tasks each slave is going to process $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$

**MaxTasks1($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. a permutation $\sigma$ and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma,\ \forall i \in [1, p] : \boxed{\sigma(i)t_{\text{com}} + c_i t_i \leq T} \right)$$

# A new optimization problem

What are we looking for?

- A permutation $\sigma$ which determines the ordering of the messages from the host: the host sends data to slave $P_i$ at time $\sigma(i)t_{\text{com}}$
- The number of tasks each slave is going to process $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$

**MaxTasks1($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. a permutation $\sigma$ and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma,\ \forall i \in [1,p] : \boxed{\sigma(i)t_{\text{com}} + c_i t_i \leq T} \right)$$

# A new optimization problem

What are we looking for?

- A permutation $\sigma$ which determines the ordering of the messages from the host: the host sends data to slave $P_i$ at time $\sigma(i)t_{\text{com}}$
- The number of tasks each slave is going to process $\mathcal{C} = \{c_1, c_2, \dots, c_p\}$

**MaxTasks1($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. a permutation $\sigma$ and an allocation $\mathcal{C} = \{c_1, c_2, \dots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max\left( \sum_{i=1}^{p} c_i \,\middle|\, \sigma,\ \forall i \in [1,p] : \boxed{\sigma(i)t_{\text{com}} + c_i t_i \leq T} \right)$$

# A new optimization problem

What are we looking for?

- A permutation $\sigma$ which determines the ordering of the messages from the host: the host sends data to slave $P_i$ at time $\sigma(i)t_{com}$
- The number of tasks each slave is going to process $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$

**MaxTasks1($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. a permutation $\sigma$ and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma,\ \forall i \in [1,p] : \boxed{c_i = \left\lfloor \frac{T - \sigma(i)t_{com}}{t_i} \right\rfloor} \right)$$

# Solution from Andonie et al. [ACGG98]

They restrict the search to allocations where the fastest processors start computing first.

They use a dynamic programming algorithm to solve the optimization problem **MinTime($C$)**.

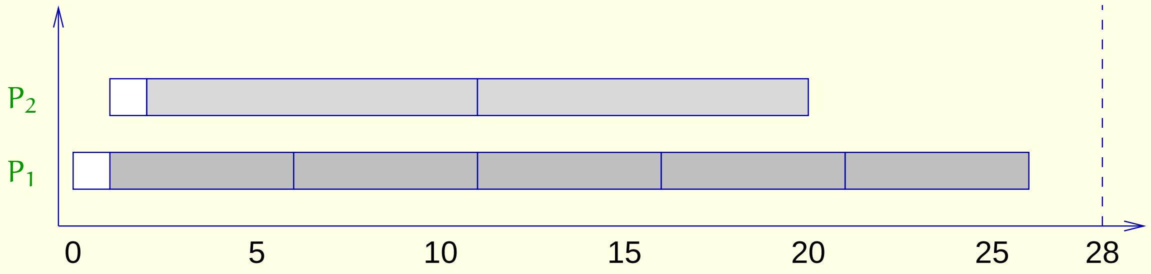
With our setting for problem **MaxTasks1($T$)**, this amount to sort the cycle-times as $t_1 \leq t_2 \leq \ldots \leq t_p$ and to let $\sigma(i) = i$ for $1 \leq i \leq p$.

The intuition is that fastest processors execute tasks more rapidly than the others, hence they should work longer.

The intuition is misleading in some cases:
$t_1 = 5$, $t_2 = 9$ and $t_{\text{com}} = 1$.

# Solution from Andonie et al. [ACGG98]

They restrict the search to allocations where the fastest processors start computing first.
They use a dynamic programming algorithm to solve the optimization problem **MinTime($C$)**.
With our setting for problem **MaxTasks1($T$)**, this amount to sort the cycle-times as $t_1 \leq t_2 \leq \ldots \leq t_p$ and to let $\sigma(i) = i$ for $1 \leq i \leq p$.
The intuition is that fastest processors execute tasks more rapidly than the others, hence they should work longer.




$\sigma = \mathrm{Id}$

7 tasks

The intuition is misleading in some cases:
$t_1 = 5$, $t_2 = 9$ and $t_{com} = 1$.

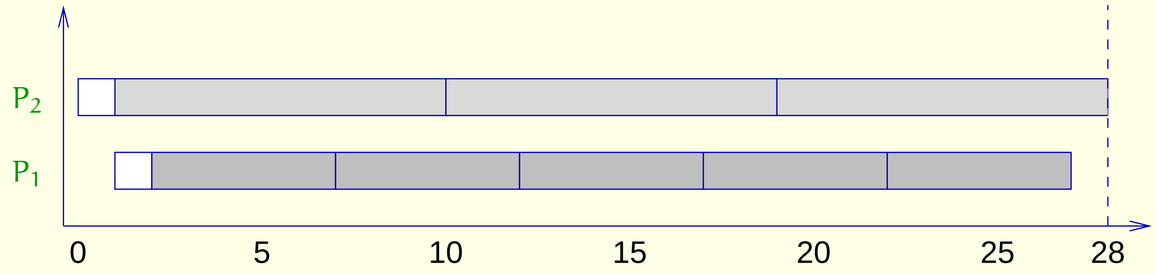
# Solution from Andonie et al. [ACGG98]

They restrict the search to allocations where the fastest processors start computing first.

They use a dynamic programming algorithm to solve the optimization problem **MinTime($C$)**.

With our setting for problem **MaxTasks1($T$)**, this amount to sort the cycle-times as $t_1 \leq t_2 \leq \ldots \leq t_p$ and to let $\sigma(i) = i$ for $1 \leq i \leq p$.

The intuition is that fastest processors execute tasks more rapidly than the others, hence they should work longer.

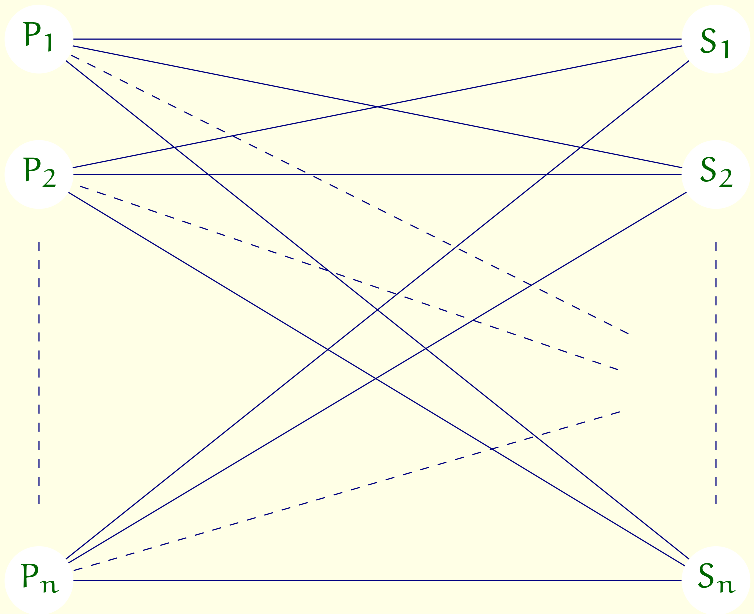



$\sigma(1) = 2, \sigma(2) = 1$

8 tasks

The intuition is misleading in some cases:
$t_1 = 5$, $t_2 = 9$ and $t_{com} = 1$.

# Optimal solution



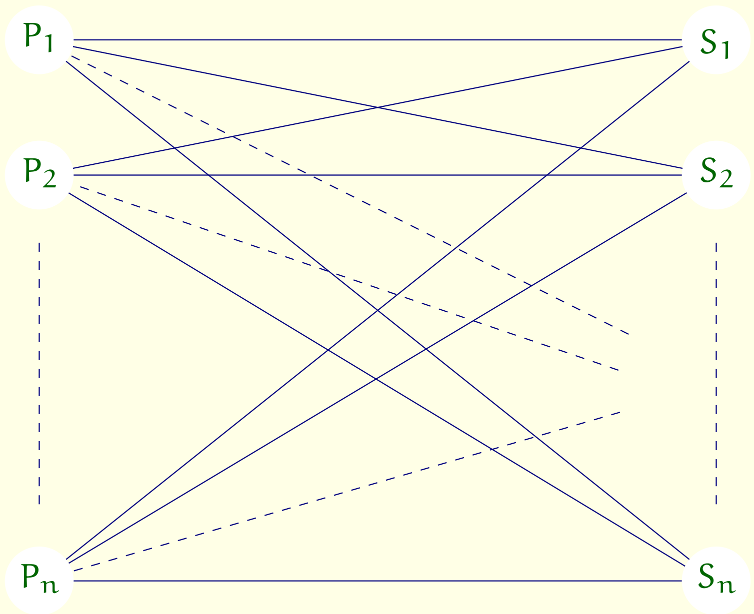

Bipartite graph for **MaxTasks1($T$).**

$$w(P_i, S_j) = \begin{cases} \text{maximum number of tasks} \\ \text{that } P_i \text{ can execute if } \sigma(i) = j \end{cases}$$
$$= \left\lfloor \frac{T - j t_{\text{com}}}{t_i} \right\rfloor$$

Solving **MaxTasks1($T$)** reduces to finding the maximum weighted matching in the bipartite graph.

**Proposition 1.** *The optimal solution to the* **MaxTasks1($T$)** *problem with initial messages can be found in time of order $O(p^3)$ with $p$ processors using some classical algorithm ([GM84, Wes96]) to find the maximum weighted matching in a bipartite graph*

# Optimal solution




Bipartite graph for **MaxTasks1($T$).**

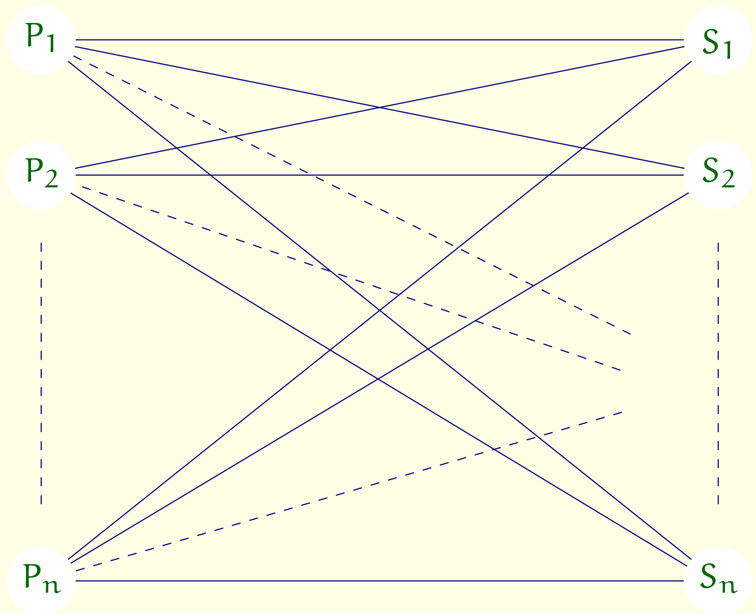
$$w(P_i, S_j) = \left\{ \begin{array}{c} \text{maximum number of tasks} \\ \text{that } P_i \text{ can execute if } \sigma(i) = j \end{array} \right.$$

$$= \left\lfloor \frac{T - jt_{com}}{t_i} \right\rfloor$$

Solving **MaxTasks1($T$)** reduces to finding the maximum weighted matching in the bipartite graph.

**Proposition 1.** *The optimal solution to the* **MaxTasks1($T$)** *problem with initial messages can be found in time of order $O(p^3)$ with $p$ processors using some classical algorithm ([GM84, Wes96]) to find the maximum weighted matching in a bipartite graph*

# Optimal solution



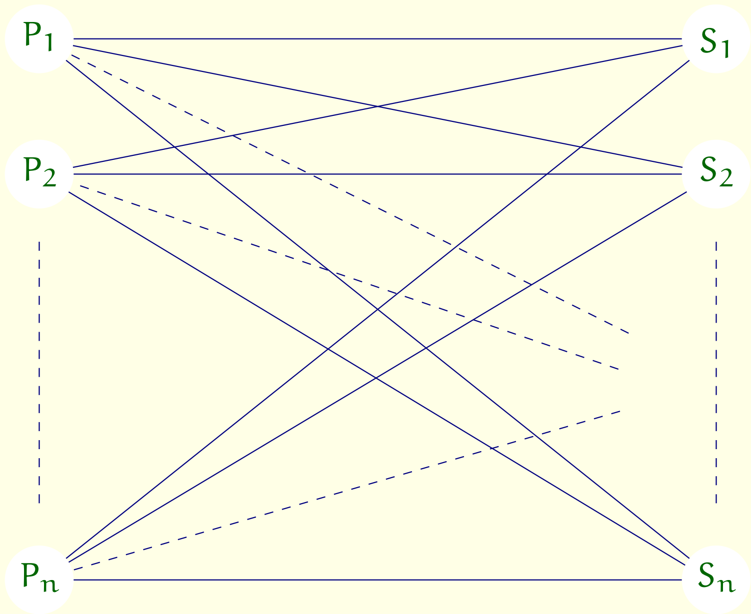

Bipartite graph for **MaxTasks1(T).**

$$w(P_i, S_j) = \left\{ \begin{array}{c} \text{maximum number of tasks} \\ \text{that } P_i \text{ can execute if } \sigma(i) = j \end{array} \right. = \left\lfloor \frac{T - jt_{\text{com}}}{t_i} \right\rfloor$$

Solving **MaxTasks1(T)** reduces to finding the maximum weighted matching in the bipartite graph.

**Proposition 1.** *The optimal solution to the* **MaxTasks1(T)** *problem with initial messages can be found in time of order* $O(p^3)$ *with* $p$ *processors using some classical algorithm ([GM84, Wes96]) to find the maximum weighted matching in a bipartite graph*

# Optimal solution




Bipartite graph for **MaxTasks1(T).**

$$w(P_i, S_j) = \begin{cases} \text{maximum number of tasks} \\ \text{that } P_i \text{ can execute if } \sigma(i) = j \end{cases}$$
$$= \left\lfloor \frac{T - j t_{\text{com}}}{t_i} \right\rfloor$$

Solving **MaxTasks1(T)** reduces to finding the maximum weighted matching in the bipartite graph.

**Proposition 1.** *The optimal solution to the* **MaxTasks1(T)** *problem with initial messages can be found in time of order* $O(p^3)$ *with* $p$ *processors using some classical algorithm ([GM84, Wes96]) to find the maximum weighted matching in a bipartite graph*

With initial and final communications

# Problem statement

Some results have been produced by the slaves or the master need some feedback on the computation:

- each slave must receive some data file from the master processor and send some results back after the processing $\rightsquigarrow$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

What are we looking for?

- A first permutation $\sigma_1$ which determines the ordering of the initial messages from the host: the host sends data to slave $P_i$ at time $\sigma_1(i)t^1_{\text{com}}$
- A second permutation $\sigma_2$ which determines the ordering of the final messages sent back to the host: given a time bound $T$, slave $P_i$ sends data back to the host at time $T - \sigma_2(i)t^2_{\text{com}}$.

# Problem statement

Some results have been produced by the slaves or the master need some feedback on the computation:

- each slave must receive some data file from the master processor and send some results back after the processing $\rightsquigarrow$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

What are we looking for?

- A first permutation $\sigma_1$ which determines the ordering of the initial messages from the host: the host sends data to slave $P_i$ at time $\sigma_1(i)t^1_{\text{com}}$
- A second permutation $\sigma_2$ which determines the ordering of the final messages sent back to the host: given a time bound $T$, slave $P_i$ sends data back to the host at time $T - \sigma_2(i)t^2_{\text{com}}$.

# Problem statement

Some results have been produced by the slaves or the master need some feedback on the computation:

- each slave must receive some data file from the master processor and send some results back after the processing $\rightsquigarrow$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

What are we looking for?

- A first permutation $\sigma_1$ which determines the ordering of the initial messages from the host: the host sends data to slave $P_i$ at time $\sigma_1(i)t^1_{\text{com}}$
- A second permutation $\sigma_2$ which determines the ordering of the final messages sent back to the host: given a time bound $T$, slave $P_i$ sends data back to the host at time $T - \sigma_2(i)t^2_{\text{com}}$.

# Problem statement

Some results have been produced by the slaves or the master need some feedback on the computation:

- each slave must receive some data file from the master processor and send some results back after the processing $\rightsquigarrow$ two communication costs: $t^1_{\mathrm{com}}$ and $t^2_{\mathrm{com}}$
- slave $P_i$ process one task in $t_i$ units of time

What are we looking for?

- A first permutation $\sigma_1$ which determines the ordering of the initial messages from the host: the host sends data to slave $P_i$ at time $\sigma_1(i)t^1_{\mathrm{com}}$
- A second permutation $\sigma_2$ which determines the ordering of the final messages sent back to the host: given a time bound $T$, slave $P_i$ sends data back to the host at time $T - \sigma_2(i)t^2_{\mathrm{com}}$.

# Problem statement

Some results have been produced by the slaves or the master need some feedback on the computation:

- each slave must receive some data file from the master processor and send some results back after the processing $\rightsquigarrow$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

What are we looking for?

- A first permutation $\sigma_1$ which determines the ordering of the initial messages from the host: the host sends data to slave $P_i$ at time $\sigma_1(i)t^1_{\text{com}}$
- A second permutation $\sigma_2$ which determines the ordering of the final messages sent back to the host: given a time bound $T$, slave $P_i$ sends data back to the host at time $T - \sigma_2(i)t^2_{\text{com}}$.

# Problem statement

Some results have been produced by the slaves or the master need some feedback on the computation:

- each slave must receive some data file from the master processor and send some results back after the processing $\rightsquigarrow$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

What are we looking for?

- A first permutation $\sigma_1$ which determines the ordering of the initial messages from the host: the host sends data to slave $P_i$ at time $\sigma_1(i)t^1_{\text{com}}$
- A second permutation $\sigma_2$ which determines the ordering of the final messages sent back to the host: given a time bound $T$, slave $P_i$ sends data back to the host at time $T - \sigma_2(i)t^2_{\text{com}}$.

# A new optimization problem

**MaxTasks2($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. two permutations $\sigma_1$ and $\sigma_2$, and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma_1, \sigma_2, \ \forall i \in [1,p] : \sigma_1(i) t_{\text{com}}^1 + c_i t_i + \sigma_2(i) t_{\text{com}}^2 \leq T \right)$$

- The solution to the MaxTasks2(T) problem with initial and final messages turns out to be surprisingly difficult.
- We do not know of any polynomial algorithm for the general case $\rightsquigarrow$ efficient guaranteed approximation using matching techniques.

# A new optimization problem

**MaxTasks2($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. two permutations $\sigma_1$ and $\sigma_2$, and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma_1, \sigma_2,\ \forall i \in [1, p] : \sigma_1(i) t^1_{\text{com}} + c_i t_i + \sigma_2(i) t^2_{\text{com}} \leq T \right)$$

- The solution to the MaxTasks2(T) problem with initial and final messages turns out to be surprisingly difficult.
- We do not know of any polynomial algorithm for the general case $\leadsto$ efficient guaranteed approximation using matching techniques.

# A new optimization problem

**MaxTasks2($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. two permutations $\sigma_1$ and $\sigma_2$, and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma_1, \sigma_2,\ \forall i \in [1,p] : \sigma_1(i) t^1_{\text{com}} + c_i t_i + \sigma_2(i) t^2_{\text{com}} \leq T \right)$$

- The solution to the MaxTasks2(T) problem with initial and final messages turns out to be surprisingly difficult.
- We do not know of any polynomial algorithm for the general case $\leadsto$ efficient guaranteed approximation using matching techniques.

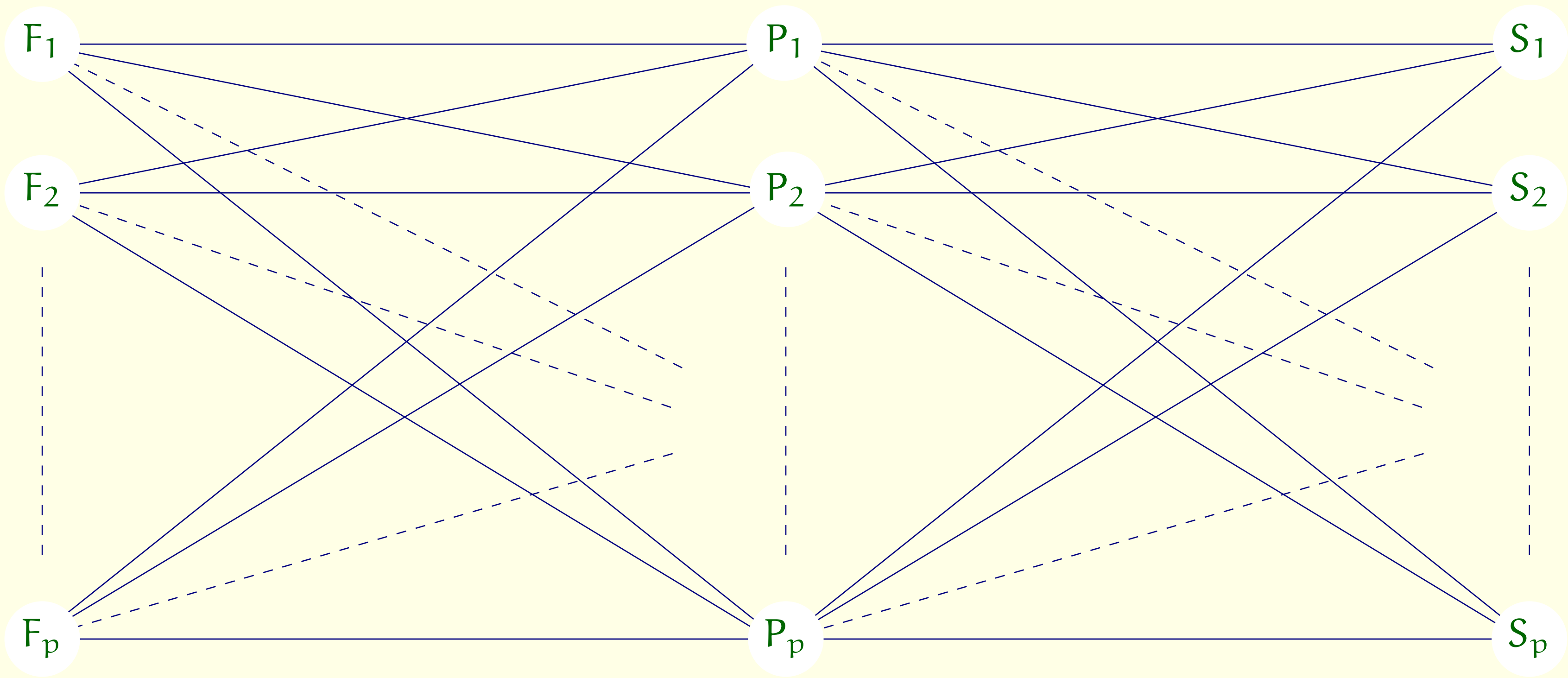
# A new optimization problem

**MaxTasks2($T$):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. two permutations $\sigma_1$ and $\sigma_2$, and an allocation $\mathcal{C} = \{c_1, c_2, \ldots, c_p\}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( \sum_{i=1}^{p} c_i \;\middle|\; \sigma_1, \sigma_2,\ \forall i \in [1,p] : \sigma_1(i) t^1_{\text{com}} + c_i t_i + \sigma_2(i) t^2_{\text{com}} \leq T \right)$$

- The solution to the MaxTasks2(T) problem with initial and final messages turns out to be surprisingly difficult.
- We do not know of any polynomial algorithm for the general case $\rightsquigarrow$ efficient guaranteed approximation using matching techniques.

# 2-factor in a Tripartite graph




Tripartite graph with initial and final communications.

$F_i$'s correspond to $\sigma_1$ and $S_i$'s correspond to $\sigma_2$. Rather than a matching, we extract a $2$-factor from the graph. The complexity of extracting $2$-factor from the graph with $3p$ vertices is of order $O(p^3)$ again.

# Edge weights approximation

The problem is that edge weights cannot be determined fully accurately; the inequality $\sigma_1(i)t^1_{\text{com}} + c_i t_i + \sigma_2(i)t^2_{\text{com}} \leq T$ translates into

$$c_i \leq \left\lfloor \frac{T - \sigma_1(i)t^1_{\text{com}} - \sigma_2(i)t^2_{\text{com}}}{t_i} \right\rfloor,$$

and we need to know both $\sigma_1(i)$ and $\sigma_2(i)$ to compute $c_i$. Instead, we use the approximation

$$\left\lfloor \frac{T/2 - \sigma_1(i)t^1_{\text{com}}}{t_i} \right\rfloor + \left\lfloor \frac{T/2 - \sigma_2(i)t^2_{\text{com}}}{t_i} \right\rfloor.$$

**Theorem 1 ([BLR01]).** *The previous approximation leads to tasks allocations that differs at most by $p$ from the optimal solution.*

# Edge weights approximation

The problem is that edge weights cannot be determined fully accurately; the inequality $\sigma_1(i)t^1_{\text{com}} + c_i t_i + \sigma_2(i)t^2_{\text{com}} \leq T$ translates into

$$c_i \leq \left\lfloor \frac{T - \sigma_1(i)t^1_{\text{com}} - \sigma_2(i)t^2_{\text{com}}}{t_i} \right\rfloor,$$

and we need to know both $\sigma_1(i)$ and $\sigma_2(i)$ to compute $c_i$. Instead, we use the approximation

$$\left\lfloor \frac{T/2 - \sigma_1(i)t^1_{\text{com}}}{t_i} \right\rfloor + \left\lfloor \frac{T/2 - \sigma_2(i)t^2_{\text{com}}}{t_i} \right\rfloor.$$

**Theorem 1 ([BLR01]).** *The previous approximation leads to tasks allocations that differs at most by $p$ from the optimal solution.*

# Edge weights approximation

The problem is that edge weights cannot be determined fully accurately; the inequality $\sigma_1(i)t^1_{\text{com}} + c_i t_i + \sigma_2(i)t^2_{\text{com}} \leq T$ translates into

$$c_i \leq \left\lfloor \frac{T - \sigma_1(i)t^1_{\text{com}} - \sigma_2(i)t^2_{\text{com}}}{t_i} \right\rfloor,$$

and we need to know both $\sigma_1(i)$ and $\sigma_2(i)$ to compute $c_i$. Instead, we use the approximation

$$\left\lfloor \frac{T/2 - \sigma_1(i)t^1_{\text{com}}}{t_i} \right\rfloor + \left\lfloor \frac{T/2 - \sigma_2(i)t^2_{\text{com}}}{t_i} \right\rfloor.$$

**Theorem 1 ([BLR01]).** *The previous approximation leads to tasks allocations that differs at most by $p$ from the optimal solution.*

# A word on complexity

Is **MaxTasks2(**$T$**)** NP-complete?

It is as difficult as the following open (polynomial vs. NP-complete) problem in combinatorial optimization (see [Hed]):

**Permutation Sums:** Let $a_1 \leq a_2 \leq \ldots \leq a_p$ be $p$ positive integers satisfying $\sum_{i=1}^{p} a_i = p(p+1)$. Do there exist two permutations $\sigma_1$ and $\sigma_2$ of $\{1, 2, \ldots, n\}$ such that

$$\forall i \in [1, p] : \sigma_1(i) + \sigma_2(i) = a_i.$$

# A word on complexity

Is **MaxTasks2($T$)** NP-complete?

It is as difficult as the following open (polynomial vs. NP-complete) problem in combinatorial optimization (see [Hed]):

**Permutation Sums:** Let $a_1 \leq a_2 \leq \ldots \leq a_p$ be $p$ positive integers satisfying $\sum_{i=1}^{p} a_i = p(p+1)$. Do there exist two permutations $\sigma_1$ and $\sigma_2$ of $\{1, 2, \ldots, n\}$ such that

$$\forall i \in [1, p] : \sigma_1(i) + \sigma_2(i) = a_i.$$

# A word on complexity

Is **MaxTasks2($T$)** NP-complete?

It is as difficult as the following open (polynomial vs. NP-complete) problem in combinatorial optimization (see [Hed]):

**Permutation Sums:** Let $a_1 \leq a_2 \leq \ldots \leq a_p$ be $p$ positive integers satisfying $\sum_{i=1}^{p} a_i = p(p+1)$. Do there exist two permutations $\sigma_1$ and $\sigma_2$ of $\{1, 2, \ldots, n\}$ such that

$$\forall i \in [1, p] : \sigma_1(i) + \sigma_2(i) = a_i.$$

With communications before each task processing

# Problem statement

Each task has its own description file:

- communications are required between the master and the slave before the processing of each task $\leadsto$ each reception costs $t_{\text{com}}$ units of time
- slave $P_i$ process one task in $t_i$ units of time

**MaxTasks3(T):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. three functions $f_{\text{startcomm}}$, $f_{\text{startcomp}}$ and $f_{\text{proc}}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max\left(N \mid \forall i \leq N,\ f_{\text{startcomp}}(i) + t_{f_{\text{proc}}(i)} \leq T\right)$$

# Problem statement

Each task has its own description file:

- communications are required between the master and the slave before the processing of each task $\rightsquigarrow$ each reception costs $t_{com}$ units of time
- slave $P_i$ process one task in $t_i$ units of time

**MaxTasks3(T):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. three functions $f_{startcomm}$, $f_{startcomp}$ and $f_{proc}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( N \mid \forall i \leq N,\ f_{startcomp}(i) + t_{f_{proc}(i)} \leq T \right)$$

# Problem statement

Each task has its own description file:

- communications are required between the master and the slave before the processing of each task $\rightsquigarrow$ each reception costs $t_{\text{com}}$ units of time
- slave $P_i$ process one task in $t_i$ units of time

**MaxTasks3(T):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. three functions $f_{\text{startcomm}}$, $f_{\text{startcomp}}$ and $f_{\text{proc}}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max\left(N \mid \forall i \leq N,\ f_{\text{startcomp}}(i) + t_{f_{\text{proc}}(i)} \leq T\right)$$

# Problem statement

Each task has its own description file:

- communications are required between the master and the slave before the processing of each task $\rightsquigarrow$ each reception costs $t_{\text{com}}$ units of time
- slave $P_i$ process one task in $t_i$ units of time

**MaxTasks3(T):** Given a time bound $T$, determine the best allocation of tasks to slaves, i.e. three functions $f_{\text{startcomm}}$, $f_{\text{startcomp}}$ and $f_{\text{proc}}$ s.t. all processors complete their execution within $T$ units of time and the total number of tasks is maximized:

$$\max \left( N \mid \forall i \leq N,\ f_{\text{startcomp}}(i) + t_{f_{\text{proc}}(i)} \leq T \right)$$

# Asymptotically optimal algorithm

Since MaxTasks3(T) is probably NP, we slightly change the problem and look for the maximum steady-state throughput of our platform.

- determining a pattern for communications and computations, that will be reproduced periodically throughout the execution
- two different cases whether the communication network is the limiting resource or not
- $\frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ represent the ratio of the communication involved by processor $P_i$
- the value of $\sum_{i=1}^{p} \frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ enable to know in which situation we are

# Asymptotically optimal algorithm

Since MaxTasks3(T) is probably NP, we slightly change the problem and look for the maximum steady-state throughput of our platform.

- determining a pattern for communications and computations, that will be reproduced periodically throughout the execution
- two different cases whether the communication network is the limiting resource or not
- $\frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ represent the ratio of the communication involved by processor $P_i$
- the value of $\sum_{i=1}^{p} \frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ enable to know in which situation we are

# Asymptotically optimal algorithm

Since MaxTasks3(T) is probably NP, we slightly change the problem and look for the maximum steady-state throughput of our platform.

- determining a pattern for communications and computations, that will be reproduced periodically throughout the execution
- two different cases whether the communication network is the limiting resource or not
- $\frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ represent the ratio of the communication involved by processor $P_i$
- the value of $\sum_{i=1}^{p} \frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ enable to know in which situation we are

# Asymptotically optimal algorithm

Since MaxTasks3(T) is probably NP, we slightly change the problem and look for the maximum steady-state throughput of our platform.

- determining a pattern for communications and computations, that will be reproduced periodically throughout the execution
- two different cases whether the communication network is the limiting resource or not
- $\frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ represent the ratio of the communication involved by processor $P_i$
- the value of $\sum_{i=1}^{p} \frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ enable to know in which situation we are

# Asymptotically optimal algorithm

Since MaxTasks3(T) is probably NP, we slightly change the problem and look for the maximum steady-state throughput of our platform.

- determining a pattern for communications and computations, that will be reproduced periodically throughout the execution
- two different cases whether the communication network is the limiting resource or not
- $\frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ represent the ratio of the communication involved by processor $P_i$
- the value of $\sum_{i=1}^{p} \frac{t_{\text{com}}}{t_{\text{com}}+t_i}$ enable to know in which situation we are

# Network is not limiting: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} \leq 1$

- All processors have to be fully used.
- $\tau_i = t_{com} + t_i$, for $1 \leq i \leq p$
- Let $T^{pattern}$ be the least common multiple of these $p$ values $\tau_i$: $T^{pattern}$ determines the length of the pattern.
- Let $\nu_i^{pattern} = \frac{T^{pattern}}{\tau_i}$

$t_{com} = 1$
$t_1 = 2$
$t_2 = 3$
$t_3 = 5$
$t_4 = 5$

# Network is not limiting: $\sum_{i=1}^{p} \frac{t_{\text{com}}}{t_{\text{com}}+t_i} \leq 1$

- All processors have to be fully used.
- $\tau_i = t_{\text{com}} + t_i$, for $1 \leq i \leq p$
- Let $T^{\text{pattern}}$ be the least common multiple of these $p$ values $\tau_i$: $T^{\text{pattern}}$ determines the length of the pattern.
- Let $\nu_i^{\text{pattern}} = \frac{T^{\text{pattern}}}{\tau_i}$

$$t_{\text{com}} = 1$$
$$t_1 = 2$$
$$t_2 = 3$$
$$t_3 = 5$$
$$t_4 = 5$$

# Network is not limiting: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} \leq 1$

- All processors have to be fully used.
- $\tau_i = t_{com} + t_i$, for $1 \leq i \leq p$
- Let $T^{pattern}$ be the least common multiple of these $p$ values $\tau_i$: $T^{pattern}$ determines the length of the pattern.
- Let $\nu_i^{pattern} = \frac{T^{pattern}}{\tau_i}$

$t_{com} = 1$

$t_1 = 2$

$t_2 = 3$

$t_3 = 5$

$t_4 = 5$

# Network is not limiting: $\sum_{i=1}^{p} \frac{t_{\text{com}}}{t_{\text{com}}+t_i} \leq 1$

- All processors have to be fully used.
- $\tau_i = t_{\text{com}} + t_i$, for $1 \leq i \leq p$
- Let $T^{\text{pattern}}$ be the least common multiple of these $p$ values $\tau_i$: $T^{\text{pattern}}$ determines the length of the pattern.
- Let $\nu_i^{\text{pattern}} = \frac{T^{\text{pattern}}}{\tau_i}$

$t_{\text{com}} = 1$

$t_1 = 2$

$t_2 = 3$

$t_3 = 5$

$t_4 = 5$

# Network is not limiting: $\sum_{i=1}^{p} \frac{t_{\text{com}}}{t_{\text{com}}+t_i} \leq 1$

- All processors have to be fully used.
- $\tau_i = t_{\text{com}} + t_i$, for $1 \leq i \leq p$
- Let $T^{\text{pattern}}$ be the least common multiple of these $p$ values $\tau_i$: $T^{\text{pattern}}$ determines the length of the pattern.
- Let $\nu_i^{\text{pattern}} = \frac{T^{\text{pattern}}}{\tau_i}$

$t_{\text{com}} = 1$

$t_1 = 2$

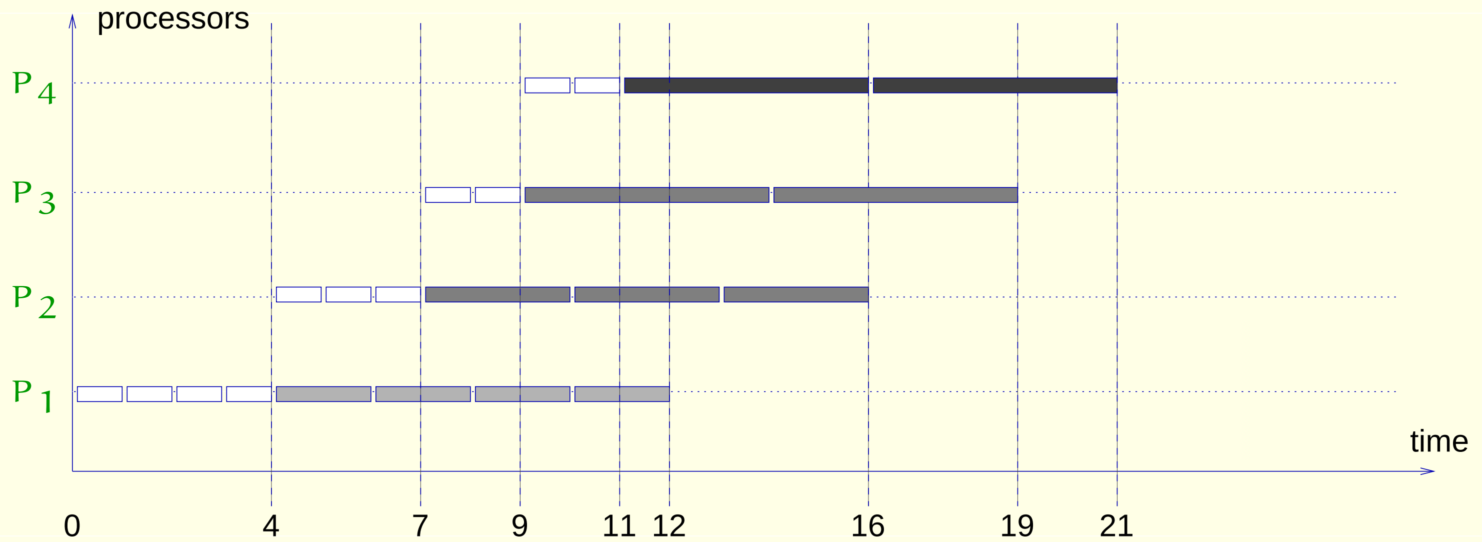
$t_2 = 3$

$t_3 = 5$

$t_4 = 5$

# Network is not limiting: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} \leq 1$

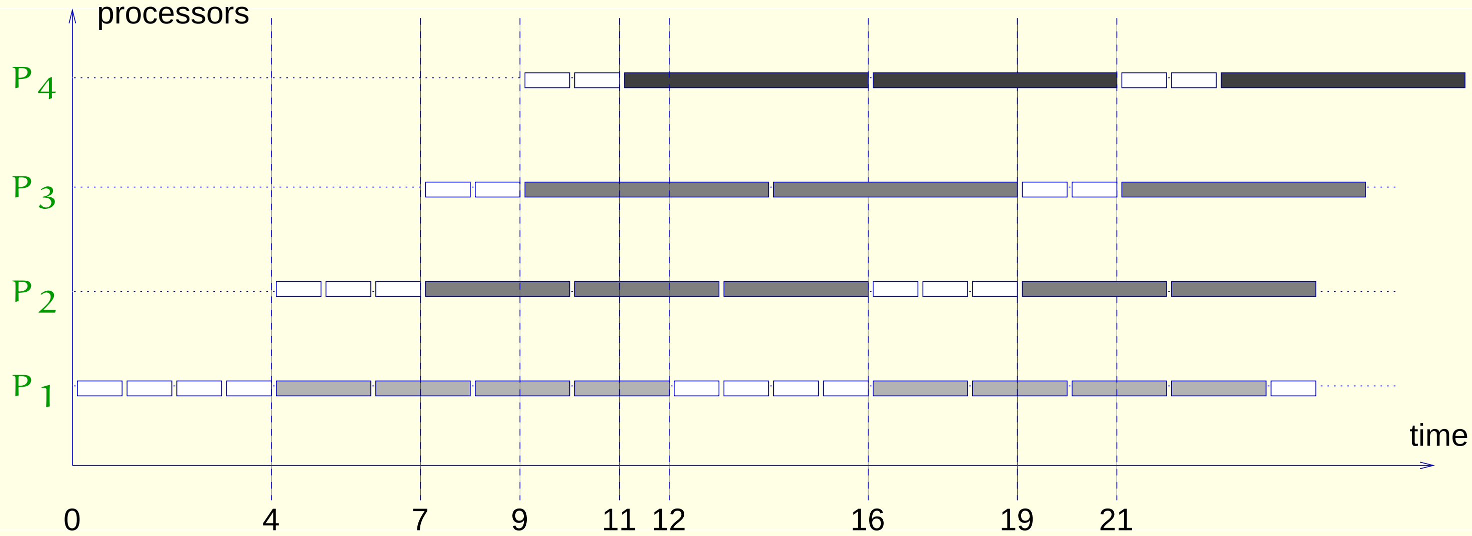
- All processors have to be fully used.
- $\tau_i = t_{com} + t_i$, for $1 \leq i \leq p$
- Let $T^{pattern}$ be the least common multiple of these $p$ values $\tau_i$: $T^{pattern}$ determines the length of the pattern.
- Let $\nu_i^{pattern} = \frac{T^{pattern}}{\tau_i}$




$t_{com} = 1$
$t_1 = 2$
$t_2 = 3$
$t_3 = 5$
$t_4 = 5$

# Network is not limiting: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} \leq 1$

- All processors have to be fully used.
- $\tau_i = t_{com} + t_i$, for $1 \leq i \leq p$
- Let $T^{pattern}$ be the least common multiple of these $p$ values $\tau_i$: $T^{pattern}$ determines the length of the pattern.
- Let $\nu_i^{pattern} = \frac{T^{pattern}}{\tau_i}$




$t_{com} = 1$

$t_1 = 2$

$t_2 = 3$

$t_3 = 5$

$t_4 = 5$

# Network is the limiting ressource: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} > 1$

- The communication link has to always be in use: some nodes will be kept idle, some will be kept busy and one processor will be partly idle.

- Sort the cycle-times of the slave processors and assume that $t_1 \leq t_2 \leq \ldots \leq t_p$. Define $\tau_i$ as before and let

$$p_{max} = \max \left\{ k \;\middle|\; \sum_{i=1}^{k} \frac{t_{com}}{t_{com} + t_i} \leq 1 \right\}.$$

- $T^{pattern}$ and the $\nu_i^{pattern}$'s are defined in the same way as before

$$t_{com} = 1$$
$$t_1 = 2$$
$$t_2 = 3$$
$$t_3 = 3$$
$$t_4 = 4$$
$$t_5 = 6$$

# Network is the limiting ressource: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} > 1$

- The communication link has to always be in use: some nodes will be kept idle, some will be kept busy and one processor will be partly idle.
- Sort the cycle-times of the slave processors and assume that $t_1 \leq t_2 \leq ... \leq t_p$. Define $\tau_i$ as before and let

$$p_{max} = \max\left\{ k \;\middle|\; \sum_{i=1}^{k} \frac{t_{com}}{t_{com}+t_i} \leq 1 \right\}.$$

- $T^{\text{pattern}}$ and the $\nu_i^{\text{pattern}}$'s are defined in the same way as before

$t_{com} = 1$

$t_1 = 2$

$t_2 = 3$

$t_3 = 3$

$t_4 = 4$

$t_5 = 6$

# Network is the limiting ressource: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} > 1$

- The communication link has to always be in use: some nodes will be kept idle, some will be kept busy and one processor will be partly idle.
- Sort the cycle-times of the slave processors and assume that $t_1 \leq t_2 \leq ... \leq t_p$. Define $\tau_i$ as before and let
$$p_{max} = \max \left\{ k \;\middle|\; \sum_{i=1}^{k} \frac{t_{com}}{t_{com} + t_i} \leq 1 \right\}.$$
- $T^{\text{pattern}}$ and the $\nu_i^{\text{pattern}}$'s are defined in the same way as before

$t_{com} = 1$

$t_1 = 2$

$t_2 = 3$

$t_3 = 3$

$t_4 = 4$

$t_5 = 6$

# Network is the limiting ressource: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} > 1$

- The communication link has to always be in use: some nodes will be kept idle, some will be kept busy and one processor will be partly idle.
- Sort the cycle-times of the slave processors and assume that $t_1 \leq t_2 \leq \ldots \leq t_p$. Define $\tau_i$ as before and let

$$p_{max} = \max\left\{ k \;\middle|\; \sum_{i=1}^{k} \frac{t_{com}}{t_{com}+t_i} \leq 1 \right\}.$$

- $T^{\text{pattern}}$ and the $\nu_i^{\text{pattern}}$'s are defined in the same way as before

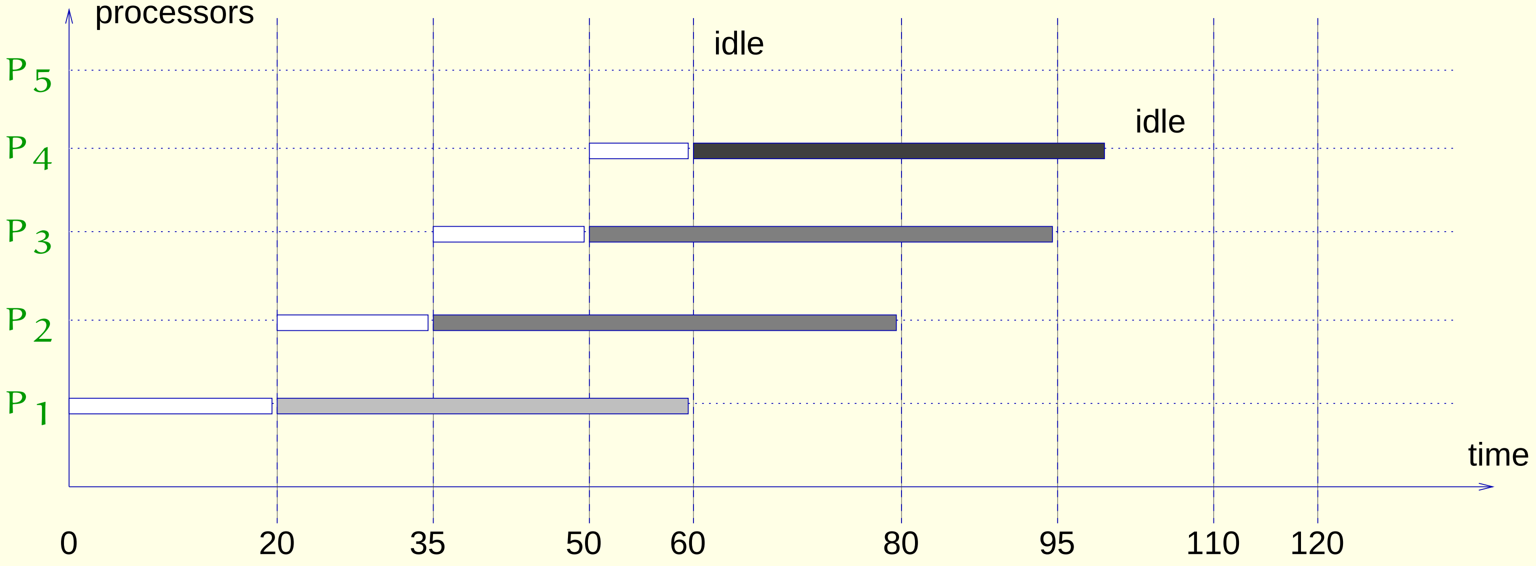
$$t_{com} = 1$$
$$t_1 = 2$$
$$t_2 = 3$$
$$t_3 = 3$$
$$t_4 = 4$$
$$t_5 = 6$$

# Network is the limiting ressource: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} > 1$

- The communication link has to always be in use: some nodes will be kept idle, some will be kept busy and one processor will be partly idle.
- Sort the cycle-times of the slave processors and assume that $t_1 \leq t_2 \leq \ldots \leq t_p$. Define $\tau_i$ as before and let

$$p_{max} = \max \left\{ k \;\middle|\; \sum_{i=1}^{k} \frac{t_{com}}{t_{com} + t_i} \leq 1 \right\}.$$

- $T^{\text{pattern}}$ and the $\nu_i^{\text{pattern}}$'s are defined in the same way as before



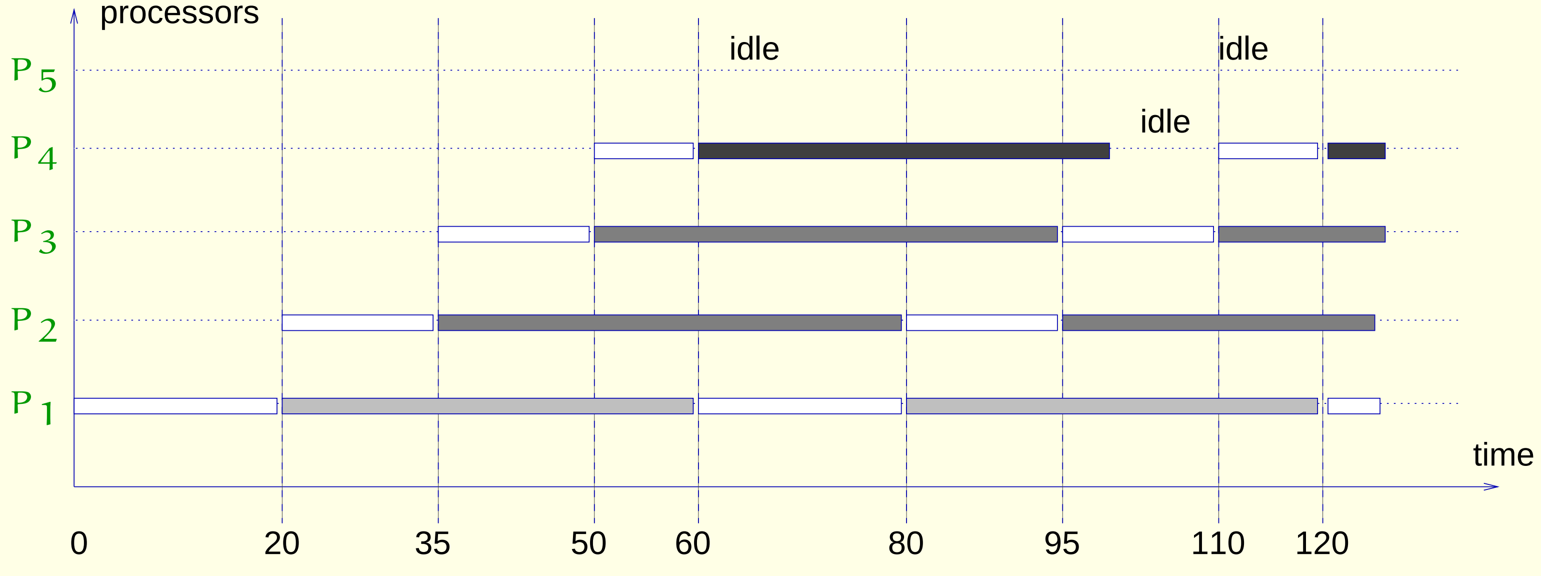

$t_{com} = 1$
$t_1 = 2$
$t_2 = 3$
$t_3 = 3$
$t_4 = 4$
$t_5 = 6$

# Network is the limiting ressource: $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} > 1$

- The communication link has to always be in use: some nodes will be kept idle, some will be kept busy and one processor will be partly idle.
- Sort the cycle-times of the slave processors and assume that $t_1 \leq t_2 \leq \ldots \leq t_p$. Define $\tau_i$ as before and let

$$p_{max} = \max \left\{ k \;\middle|\; \sum_{i=1}^{k} \frac{t_{com}}{t_{com} + t_i} \leq 1 \right\}.$$

- $T^{pattern}$ and the $\nu_i^{pattern}$'s are defined in the same way as before




$t_{com} = 1$

$t_1 = 2$

$t_2 = 3$

$t_3 = 3$

$t_4 = 4$

$t_5 = 6$

# Asymptotical optimality

**Theorem 2 ([BLR01]).** *Let* $N_{opt}(T)$ *be the optimal number of tasks that can be executed within* $T$ *time-steps. Let* $N(T)$ *be the number of tasks executed by the first algorithm if* $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} \leq 1$*, and by the second algorithm if* $\sum_{i=1}^{p} \frac{t_{com}}{t_{com}+t_i} > 1$*. Then*

$$\lim_{T \to \infty} \frac{N(T)}{N_{opt}(T)} = 1.$$

With communications both before and after each task processing

# Problem statement

- communications are required between the master and the slave both before and after the processing of each task $\rightsquigarrow$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

Use the previous algorithm with $t_{\text{com}} = t^1_{\text{com}} + t^2_{\text{com}}$

**Theorem 3.** *Let $N_{opt}(T)$ be the optimal number of tasks that can be executed within $T$ time-steps, and let $N(T)$ be the number of tasks executed by our algorithm. Then*

$$\lim_{T \to \infty} \frac{N(T)}{N_{opt}(T)} = 1.$$

# Problem statement

- communications are required between the master and the slave both before and after the processing of each task $\rightsquigarrow$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

Use the previous algorithm with $t_{\text{com}} = t^1_{\text{com}} + t^2_{\text{com}}$

**Theorem 3.** *Let $N_{opt}(T)$ be the optimal number of tasks that can be executed within $T$ time-steps, and let $N(T)$ be the number of tasks executed by our algorithm. Then*

$$\lim_{T\to\infty} \frac{N(T)}{N_{opt}(T)} = 1.$$

# Problem statement

- communications are required between the master and the slave both before and after the processing of each task $\rightsquigarrow$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

Use the previous algorithm with $t_{\text{com}} = t^1_{\text{com}} + t^2_{\text{com}}$

**Theorem 3.** *Let $N_{opt}(T)$ be the optimal number of tasks that can be executed within $T$ time-steps, and let $N(T)$ be the number of tasks executed by our algorithm. Then*

$$\lim_{T\to\infty} \frac{N(T)}{N_{opt}(T)} = 1.$$

# Problem statement

- communications are required between the master and the slave both before and after the processing of each task $\leadsto$ two communication costs: $t^1_{\text{com}}$ and $t^2_{\text{com}}$
- slave $P_i$ process one task in $t_i$ units of time

Use the previous algorithm with $t_{\text{com}} = t^1_{\text{com}} + t^2_{\text{com}}$

**Theorem 3.** *Let $N_{opt}(T)$ be the optimal number of tasks that can be executed within $T$ time-steps, and let $N(T)$ be the number of tasks executed by our algorithm. Then*

$$\lim_{T \to \infty} \frac{N(T)}{N_{opt}(T)} = 1.$$

# Related Work

- the most related work is presented in the paper of Andonie et al [ACGG98]
- parallel machine problems with a server : several complexity (NP-completeness) results [HPS00, KW97, BDFK$^+$00] and providing guaranteed approximations [LG96] **MaxTasks1(**T**)** is a very special instance of this class of server problems
- **MaxTasks2(**T**)** is a special instance of the job-shop scheduling problem (see problem SS18 in [GJ91]).Because this instance is very specific, we do not know its complexity (polynomial versus NP-complete).
- **MaxTasks3** and **MaxTasks4** have been extended to trees in [BCF$^+$01]: collaborative work with Larry Carter and Jeanne Ferrante (UCSD)

# Related Work

- the most related work is presented in the paper of Andonie et al [ACGG98]
- parallel machine problems with a server : several complexity (NP-completeness) results [HPS00, KW97, BDFK$^+$00] and providing guaranteed approximations [LG96] **MaxTasks1(**T**)** is a very special instance of this class of server problems
- **MaxTasks2(**T**)** is a special instance of the job-shop scheduling problem (see problem SS18 in [GJ91]).Because this instance is very specific, we do not know its complexity (polynomial versus NP-complete).
- **MaxTasks3** and **MaxTasks4** have been extended to trees in [BCF$^+$01]: collaborative work with Larry Carter and Jeanne Ferrante (UCSD)

# Related Work

- the most related work is presented in the paper of Andonie et al [ACGG98]
- parallel machine problems with a server : several complexity (NP-completeness) results [HPS00, KW97, BDFK$^{+}$00] and providing guaranteed approximations [LG96] **MaxTasks1(**$\mathcal{T}$**)** is a very special instance of this class of server problems
- **MaxTasks2(**$\mathcal{T}$**)** is a special instance of the job-shop scheduling problem (see problem SS18 in [GJ91]).Because this instance is very specific, we do not know its complexity (polynomial versus NP-complete).
- **MaxTasks3** and **MaxTasks4** have been extended to trees in [BCF$^{+}$01]: collaborative work with Larry Carter and Jeanne Ferrante (UCSD)

# Related Work

- the most related work is presented in the paper of Andonie et al [ACGG98]
- parallel machine problems with a server : several complexity (NP-completeness) results [HPS00, KW97, BDFK$^+$00] and providing guaranteed approximations [LG96] **MaxTasks1(T)** is a very special instance of this class of server problems
- **MaxTasks2(T)** is a special instance of the job-shop scheduling problem (see problem SS18 in [GJ91]).Because this instance is very specific, we do not know its complexity (polynomial versus NP-complete).
- **MaxTasks3** and **MaxTasks4** have been extended to trees in [BCF$^+$01]: collaborative work with Larry Carter and Jeanne Ferrante (UCSD)

# Related Work

Our four problems differ from those studied in the literature with a server and start-up times in that

- all tasks are identical and independent,
- communication times (the counterpart of the set-up times) are identical too.

The difficulty lies solely in the heterogeneity of the computing resources.

# Conclusion

- Deriving efficient algorithms for the simple master-slave paradigm, in the framework of heterogeneous computing resources, turns out to be surprisingly difficult.
- Optimal polynomial algorithm in the case of an initial scattering of data.
- Guaranteed polynomial approximation algorithm in the case of initial and final communications. We conjecture this problem to be intrinsically difficult even on (intuitively) simple instances.
- Asymptotically optimal algorithms for the case where each task processing must be preceded (and possibly followed) by a communication from (back to) the master

# Conclusion

- Deriving efficient algorithms for the simple master-slave paradigm, in the framework of heterogeneous computing resources, turns out to be surprisingly difficult.
- Optimal polynomial algorithm in the case of an initial scattering of data.
- Guaranteed polynomial approximation algorithm in the case of initial and final communications. We conjecture this problem to be intrinsically difficult even on (intuitively) simple instances.
- Asymptotically optimal algorithms for the case where each task processing must be preceded (and possibly followed) by a communication from (back to) the master

# Conclusion

- Deriving efficient algorithms for the simple master-slave paradigm, in the framework of heterogeneous computing resources, turns out to be surprisingly difficult.
- Optimal polynomial algorithm in the case of an initial scattering of data.
- Guaranteed polynomial approximation algorithm in the case of initial and final communications. We conjecture this problem to be intrinsically difficult even on (intuitively) simple instances.
- Asymptotically optimal algorithms for the case where each task processing must be preceded (and possibly followed) by a communication from (back to) the master

# Conclusion

- Deriving efficient algorithms for the simple master-slave paradigm, in the framework of heterogeneous computing resources, turns out to be surprisingly difficult.
- Optimal polynomial algorithm in the case of an initial scattering of data.
- Guaranteed polynomial approximation algorithm in the case of initial and final communications. We conjecture this problem to be intrinsically difficult even on (intuitively) simple instances.
- Asymptotically optimal algorithms for the case where each task processing must be preceded (and possibly followed) by a communication from (back to) the master

## References

[ACGG98] R. Andonie, A.T. Chronopoulos, D. Grosu, and H. Galmeanu. Distributed backpropagation neural networks on a PVM heterogeneous system. In Parallel and Distributed Computing and Systems Conference (PDCS'98), pages 555–560. IASTED Press, 1998.

[BCF+01] Olivier Beaumont, Larry Carter, Jeanne Ferrante, Arnaud Legrand, and Yves Robert. Bandwidth-centric allocation of independent taks on heterogeneous platorms. Technical Report RR-2001-25, LIP, ENS Lyon, 2001. Available at `www.ens-lyon.fr/LIP/`.

[BDFK+00] P. Brucker, C. Dhaenens-Flipo, S. Knust, S.A. Kravchenko, and F. Werner. Complexity results for parallel machine problems with

# Bibliography


a single server. Technical Report Reihe P, No. 219, Fachbereich Mathematik Informatik, Universität Osnabrück, 2000.

[BLR01] Olivier Beaumont, Arnaud Legrand, and Yves Robert. The master-slave paradigm with heterogeneous processors. Technical Report RR-2001-13, LIP, ENS Lyon, March 2001. Available at `www.ens-lyon.fr/LIP/`.

[GJ91] Michael R. Garey and Davis S. Johnson. Computers and Intractability, a Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1991.

[GM84] M. Gondran and M. Minoux. Graphs and Algorithms. John Wiley & Sons, 1984.

[Hed] Steve Hedetniemi. Open Problems in Combinatorial Optimization. World Wide Web document, URL: `http://www.cs.clemson.edu/~hedet/algorithms.html`.

# Bibliography


[HPS00] N. Hall, C.N. Potts, and C. Sriskandarajah. Parallel machine scheduling with a common server. Discrete Applied Mathematics, 102:223–243, 2000.

[KW97] S.A. Kravchenko and F. Werner. Parallel machine scheduling problems with a single server. Mathematical Computational Modelling, 26:1–11, 1997.

[LG96] H. Lee and M. Guignard. A hybrid bounding procedure for the workload allocation problem on parallel unrelated machines with setups. Journal of the Operational Research Society, 47:1247–1261, 1996.

[SBSS98] J.R. Stiles, T.M. Bartol, M.M. Salpeter, and M.M. Salpeter. Monte Carlo simulation of neuromuscular transmitter release using MCell, a general simulator of cellular physiological processes. Computational Neuroscience, pages 279–284, 1998.

# Bibliography


[SFM+] Kizhake Soman, Robert Fraczkiewicz, Christian Mumenthaler, Berthold von Freyberg, and Thomas Schaumannand Werner Braun. FANTOM - (Fast Newton - Raphson Torsion Angle Minimizer). World Wide Web document, URL: `http://www.scsb.utmb.edu/fantom/fm_home.html`. a program for "the calculation of conformations of linear and cyclic polypeptides and proteins with low conformational energies including distance and dihedral angle constraints from nuclear magnetic resonance experiments or for modeling purposes.".

[Wes96] D.B. West. *Introduction to Graph Theory*. Prentice Hall, 1996.

[WT98] J. Watts and S. Taylor. A practical approach to dynamic load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 9(3):235–248, 1998.