# Independent Tasks Scheduling on Heterogeneous Platforms under Bounded Multi-Port Model

Olivier Beaumont, Nicolas Bonichon, Lionel Eyraud-Dubois
INRIA Bordeaux Sud-Ouest, CEPAGE, LaBRI

Loris Marchal
CNRS, ENS Lyon, ROMA, LIP

Scheduling in Aussois
June 2011

# Outline

# Outline

# Communication Models in the literature

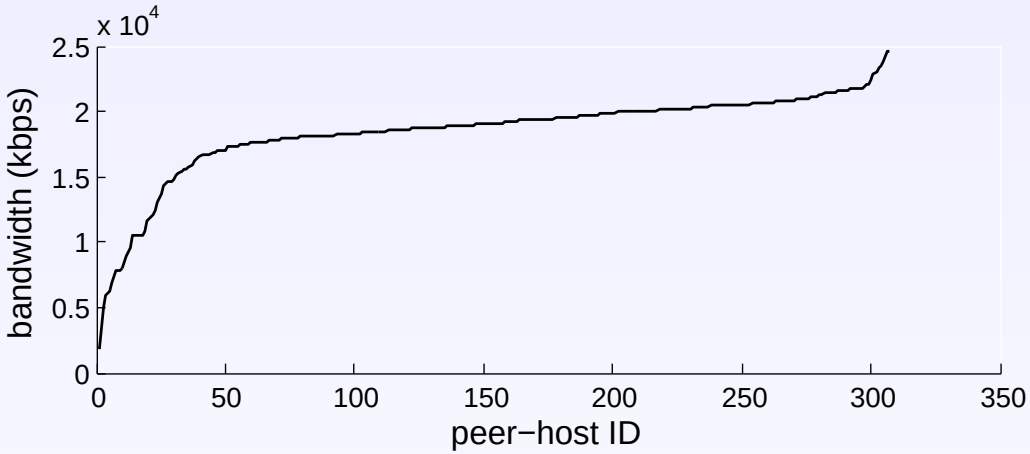
- No contention
- One-port
  - a node can be involved in at most one communication
  - comes into two flavors (unidirectional or bidirectional)
  - associated to a topology (physical or at application level)
- Multi-port
  - a node can be involved in several communications
  - provided that incoming and outgoing bandwidths are not exceeded
  - associated to an overlay network

# Topology vs Coordinate Systems

- ☺ Topology is more precise
  - but is not known in general
  - and tools to discover topology (ENV, AINEM) are too slow
  - especially if the churn is high !
- ☺ Coordinate systems
  - embed the nodes into a metric space
  - *i.e.* give coordinates to them
  - and use their coordinates to approximate the available bandwidth (or the latency) between them.
  - Examples : Vivaldi (2D+1), Sequoia (Trees), PathGuru (traceroute), DMF (SVD), LastMile (bin, bout)

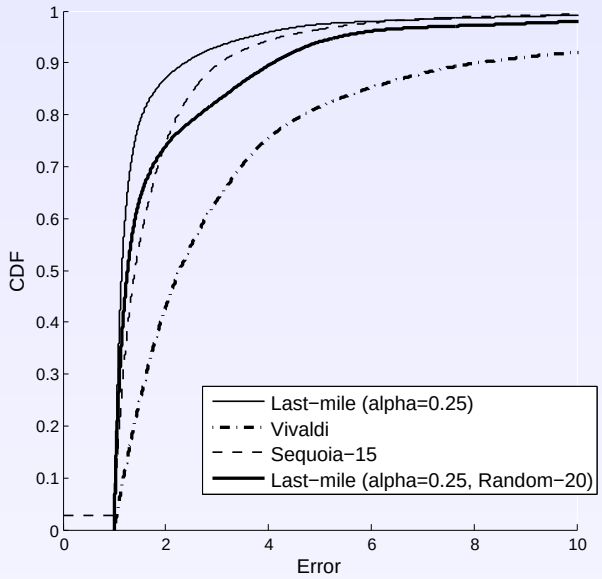
# Comparison of embedding tools [B, Eyraud-Dubois, Won, Europar'11]

- Extensive comparison of embedding tools
- On actual PlanetLab data
- Outgoing bandwidth for a Planetlab node typically looks like this




- at first, limited by other nodes incoming bandwidths
- then by its own outgoing bandwidth
- right part : nodes in the same local network, bad measurements ?
- $b_i^{\text{OUT}} \leftrightarrow$ height of the flat area.
- $\textsc{Bandwidth}(P_i, P_j) = \min(b_i^{\text{OUT}}, b_j^{\text{IN}})$

# Comparison of embedding tools (2)

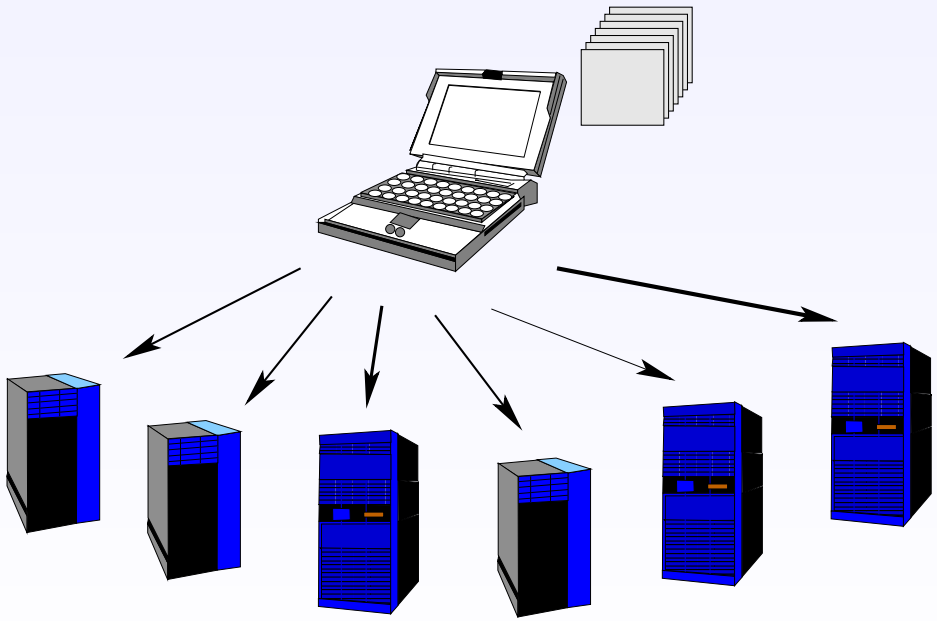



- $\text{ERROR} = \max(\frac{\text{ESTIMATED}}{\text{MEASURED}}, \frac{\text{MEASURED}}{\text{ESTIMATED}})$
- $(x, y)$: ERROR is less than $x$ for a fraction $y$ of pairs
- Conclusion : LastMile for estimating $\text{BANDWIDTH}(P_i, P_j)$ is
  - cheap and robust and decentralized
  - at least as precise
  - for PlanetLab dataset (and probably even better for DSL nodes)

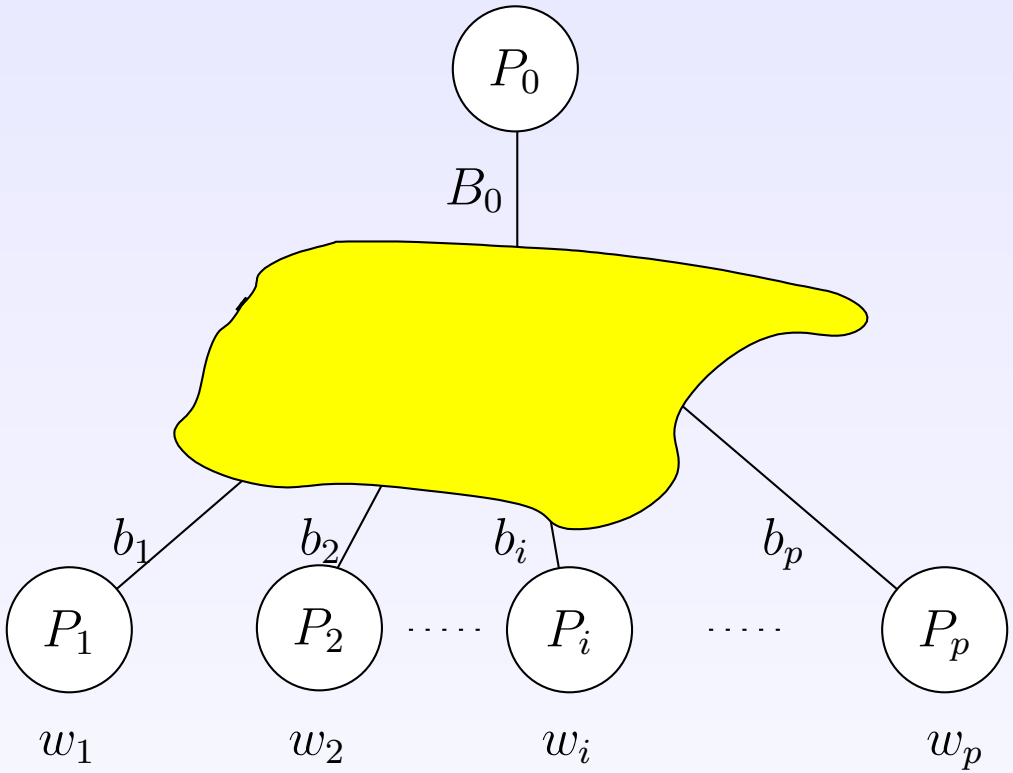
# Outline

# Divisible Load

- One master, holding a large number of identical tasks, $P$ workers
- Heterogeneity in computing speed and bandwidth
- Master holding $N$ tasks
- Worker $P_i$ will get a fraction $X_i = \alpha_i \times N$ of these tasks
- $\alpha_i$ is **rational**, tasks are divisible
- $\Rightarrow$ possible to derive analytical solutions (tractability)

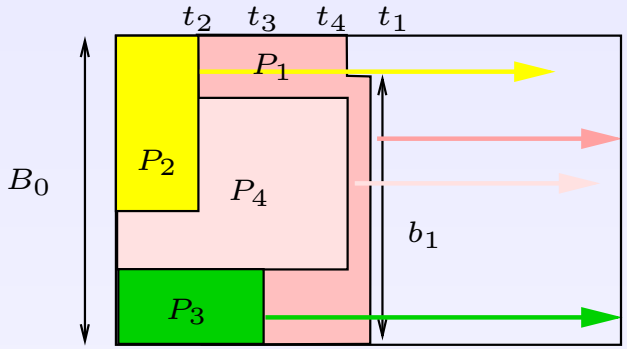


# Bounded Multi-Port Model



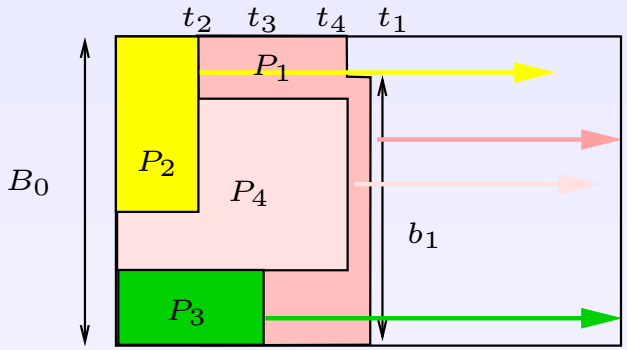

- $B_0$: output bandwidth of the master processor.
- $b_i$: input bandwidth of $P_i$.
- $w_i$: time to process a unit size task on $P_i$.
- Use of QoS mechanisms to achieve prescribed bandwidth sharing.
  - between $(P_0, P_i)$ and $(P_0, P_j)$.

# Bounded Multi-Port Model (1)




- Notations:
  - $b'_i(t)$: actual bandwidth used at time $t$ by the communication between $P_0$ and $P_i$.
  - $t_i$: the time when processor $P_i$ stops communicating.
  - $X_i$: the fractional number of tasks sent to $P_i$.
- Constraints:
  - input bandwidth: $\forall t, b'_i(t) \leqslant b_i$.
  - output bandwidth: $\forall t, \sum_i b'_i(t) \leqslant B_0$.

# Bounded Multi-Port Model (2)



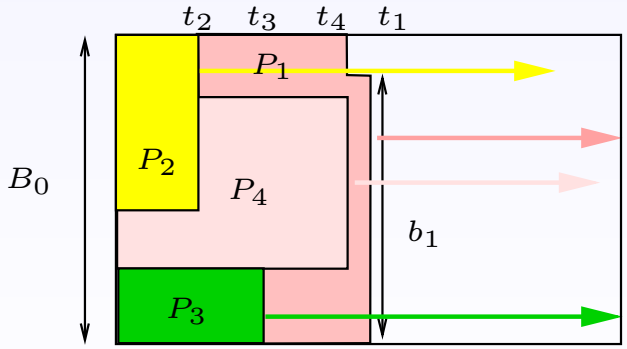

- Processing constraints :
  - 1rst part: With a linear cost model $X_i$ units of work:
    - ★ sent to $P_i$ in $\frac{X_i}{\int_t b_i'(t)}$ time units
    - ★ processed by $P_i$ in $X_i \times w_i$ time units
  - 2nd part: With an affine cost model: code of size $S_i$ + data $X_i$ units of work
    - ★ sent to $P_i$ in $\frac{S_i+X_i}{\int_t b_i'(t)}$ time units
    - ★ processed by $P_i$ in $X_i \times w_i$ time units

# Normal Form

## Definition

A schedule is said to be in *normal form* if

1. all processors are involved in the processing of tasks,
2. all slaves start processing tasks immediately after the end of the communication with the master (at time $t_i$) and stop processing at time 1,
3. during each time slot $]t_{i_k}, t_{i_{k+1}}]$ the bandwidth used by any processor is constant.

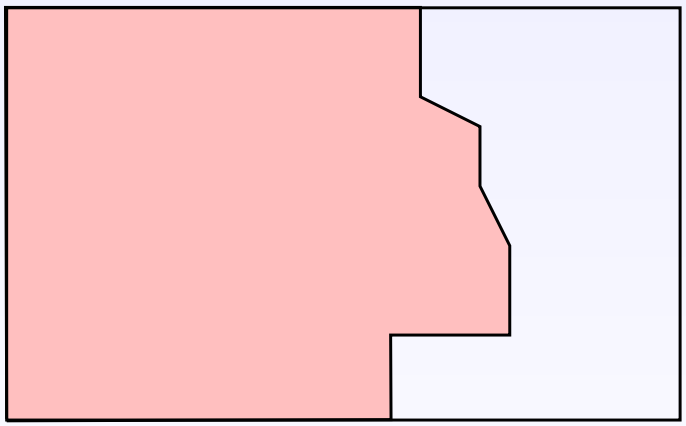


# Lemma 1

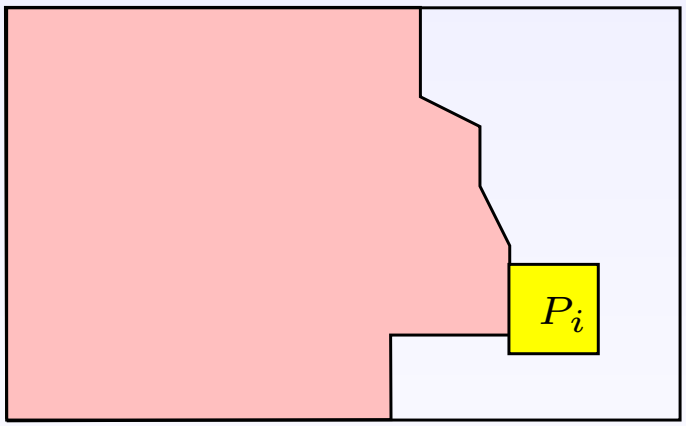
**Lemma**

*In an optimal schedule, all processors take part in the computations.*

*Proof:*



Original schedule



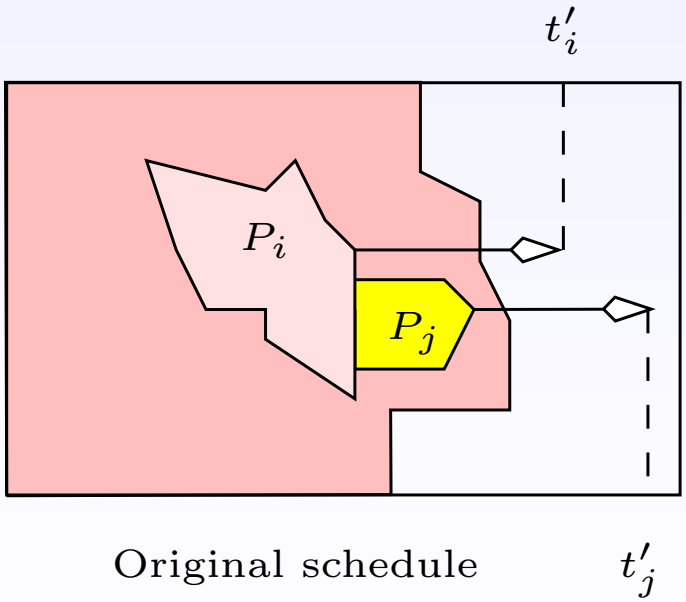

New schedule

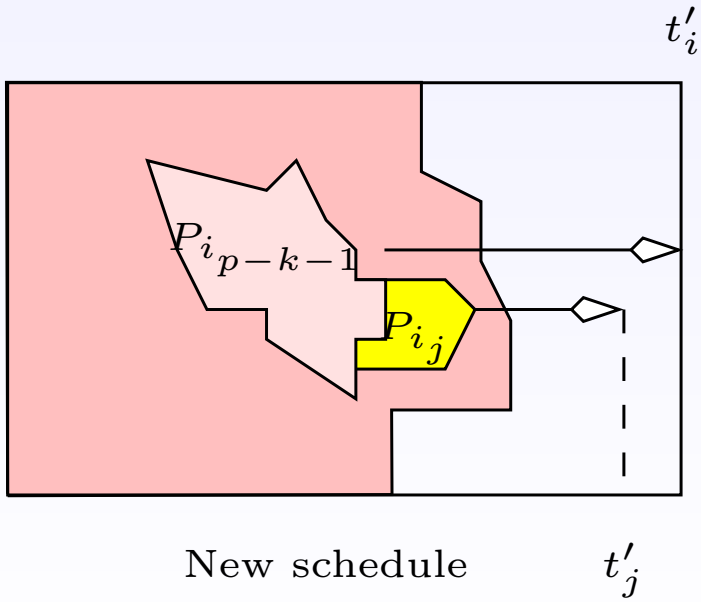
# Lemma 2

## Lemma

*In an optimal schedule, slaves start processing tasks immediately after the end of the communication with the master and stop processing at time 1, i.e. there is no idle time between the end of the communication and the deadline.*

*Proof:* By induction on the first slave that stop processing before $t = 1$.



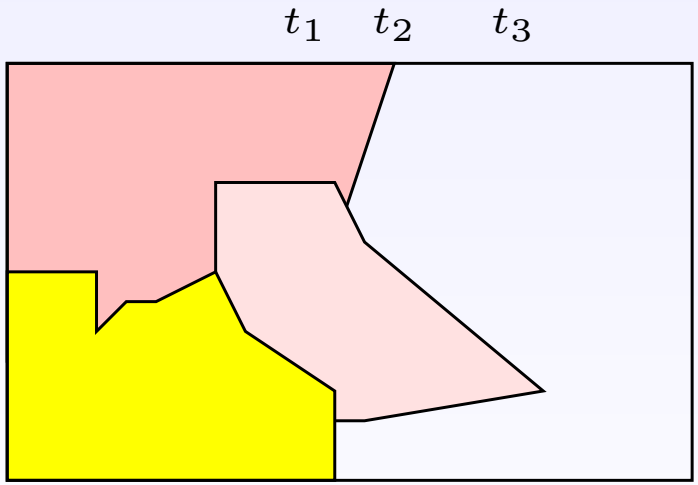



# Lemma 3

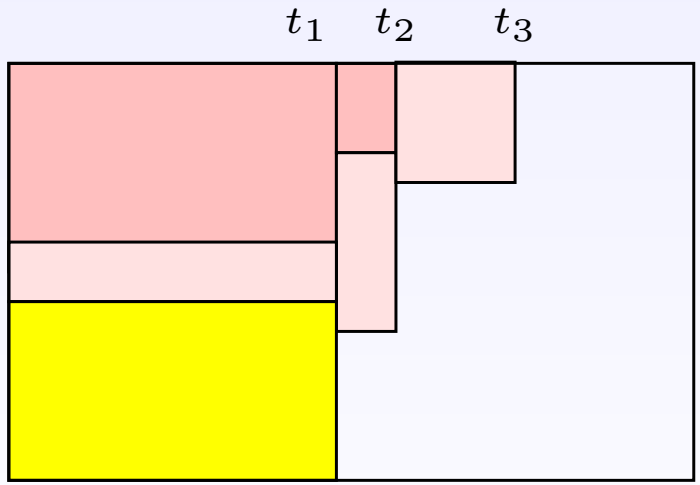
## Lemma

*There exists an optimal schedule such that during each time slot* $]t_{i_k}, t_{i_{k+1}}]$ *the bandwidth used by any processor is constant.*

*Proof:*




Original schedule




New schedule

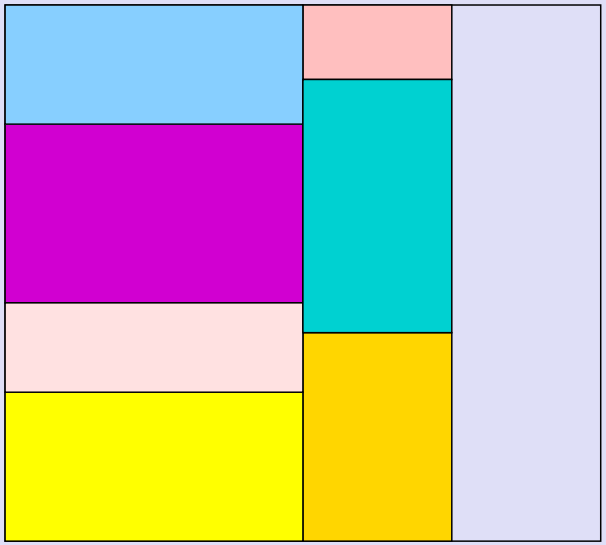
## Theorem

BoundedMPDivisible *is NP-complete.*

## Proof.

Reduction from 2-Partition. We build an instance of BoundedMPDivisible with $w_i = 1/b_i$ and $\sum_i b_i = 2B_0$.

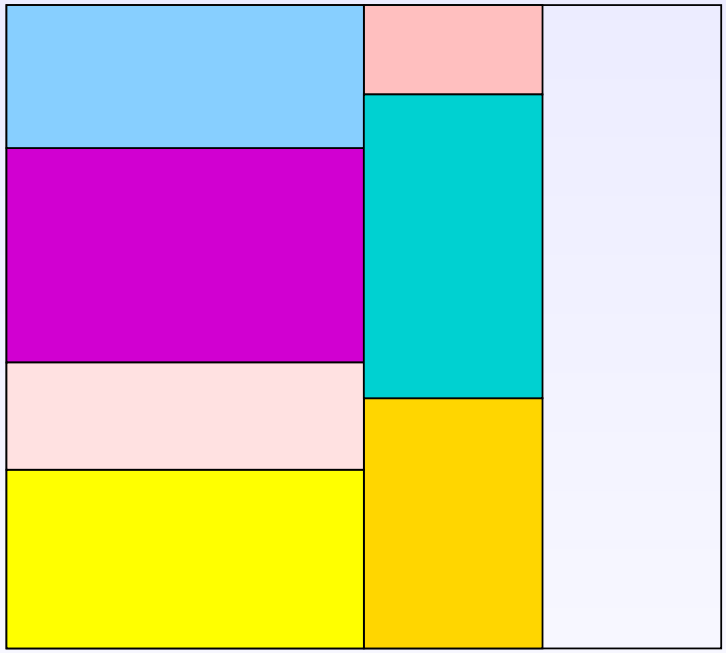
- If $P_i$ receives data at rate $b_i$ during $t$ time units
- it will take $b_i w_i t = t$ time units to process them.



$Q = 3B_0/4$

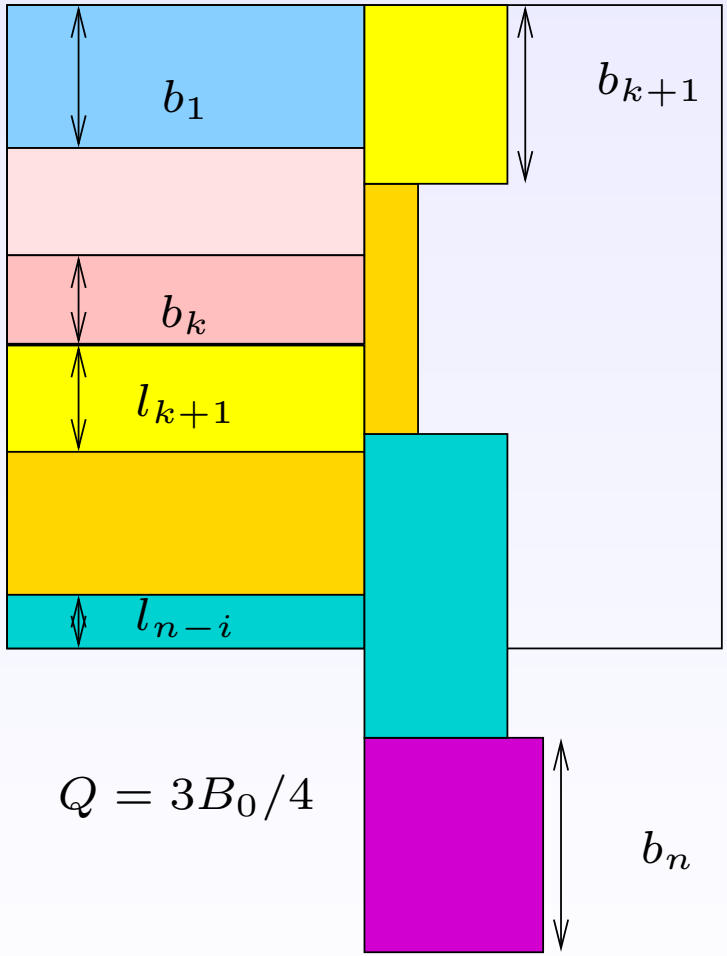
# NP-Completeness: proof

(a) Optimal solution



$Q = 3B_0/4$

(b) Hypothetic solution




$Q = 3B_0/4$

# Robustness Issues

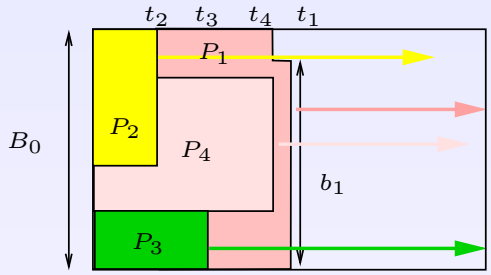
- Finding the optimal ordering under 1-port model is easy
- but a bad guess may lead to arbitrarily bad results
  - Consider $P_1 = (1, 1)$ and $P_2 = (\frac{1}{\varepsilon}, 1)$
  - if $P_1$ is scheduled first: $\frac{1+\varepsilon}{2}$ tasks
  - if $P_2$ scheduled first: $\frac{3\varepsilon}{2}$ tasks only!
- Under bounded multi-port model,
  - finding the optimal schedule is more difficult (NP-Complete)
  - but the worst ratio is $\frac{8}{9}$ (proved) between priority-based schedules (in the case of 2 processors)!
  - and we conjecture that it is even better in the case $P > 2$ !

what may be useful if bandwidth estimation are unreliable!

- In the case of BMP, the "almost only" important thing is to correctly estimate processing time to stop the communications at the right time!

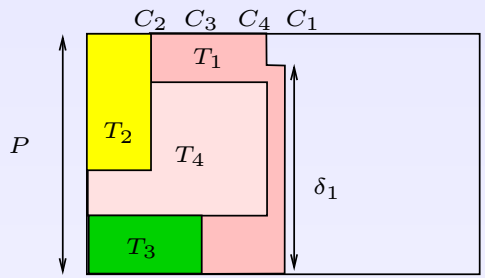
# Outline

# Bounded Multi-Port Model – Fixed Communication Cost




- Notations:
  - $b'_i(t)$: actual quantity of resources used at time $t$ by the communication between $P_0$ and $P_i$,
  - $t_i$: the time when processor $P_i$ stops communicating,
  - $q_i$: the fractional number of tasks sent to $P_i$.
- Processing constraints : affine cost model
  - $S_i$ size of the code ($+$ $X_i$ very small) units of work,
  - sent to $P_i$ in $\frac{S_i}{\int_t b'_i(t)}$ time units,
  - computed by $P_i$ in $X_i \times w_i$ time units

# Bounded Multi-Port Model – Malleable Tasks Scheduling




- Notations:
  - Task $T_i$ with maximum degree of parallelism $\delta_i$ and size $S_i$
  - $S_i$ denotes the overall cost, does not depend on the nb of procs
  - $b'_i(t)$ actual quantity of resources allocated to $T_i$ at time $t$
  - $C_i$: completion time of $T_i$, $\int_0^{C_i} b'_i(t) = S_i$.
- Weighted sum of completion times
  - Goal: $\textsc{Maximize} \sum(1 - C_i)w_i \iff \textsc{Minimize} \sum_i w_i C_i$
  - very similar to divisible load setting when minimizing the weighted sum of completion times
  - except that the quantity of allocated resource should be integers (Ok, B/s too)

# Complexity Results (1)

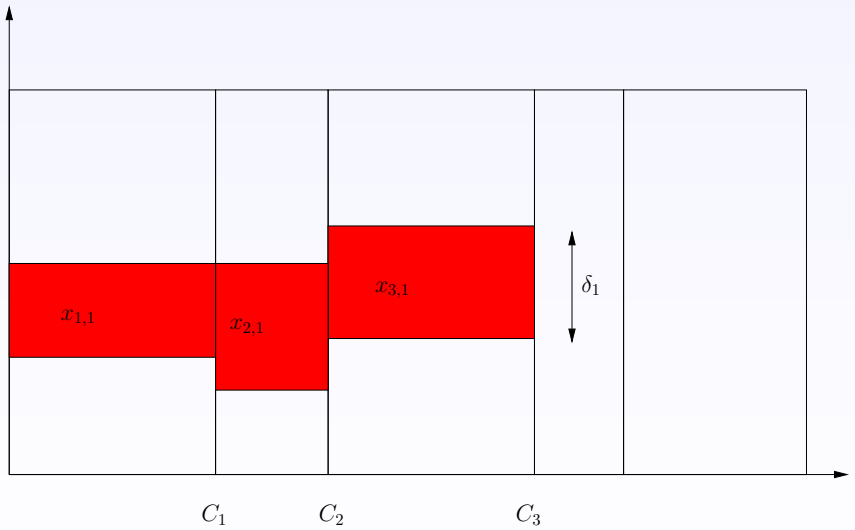
- $P|pmtn; var; p_i(q) = p_i/q, \delta_i| \sum w_i C_i$ is NP Complete
- Proof: Generalization of $P|pmtn| \sum w_i C_i$.
- Remark: Makespan versions are much easier
  - $P|var; p_i(q) = p_i/q, \delta_i, r_i|C_{\max}$ (even in presence of release dates) solvable in time $O(n^2)$
  - Maximum lateness problem $P|var; p_i(q) = p_i/q, \delta_i, r_i|L_{\max}$ solvable in time $O(n^4 P)$
- Results are known for the sum of completion times
  - Schwiegelshohn: 2.37-approx without preemptions but suspensions allowed
  - Kawaguchi and Kyan: $\frac{1+\sqrt{2}}{2}$-approx (with or without preemptions) when $\delta_i = 1$.
  - Deng et al.: 2-approximation algorithm (varying nb of resources) in the online and non-clairvoyant case.

# Complexity Results (2)

- $C_i$ = completion time of $T_i$. Assume $C_1 \leqslant C_2 \leqslant \ldots \leqslant C_n$.
- Then, the following LP provides the optimal solution
-

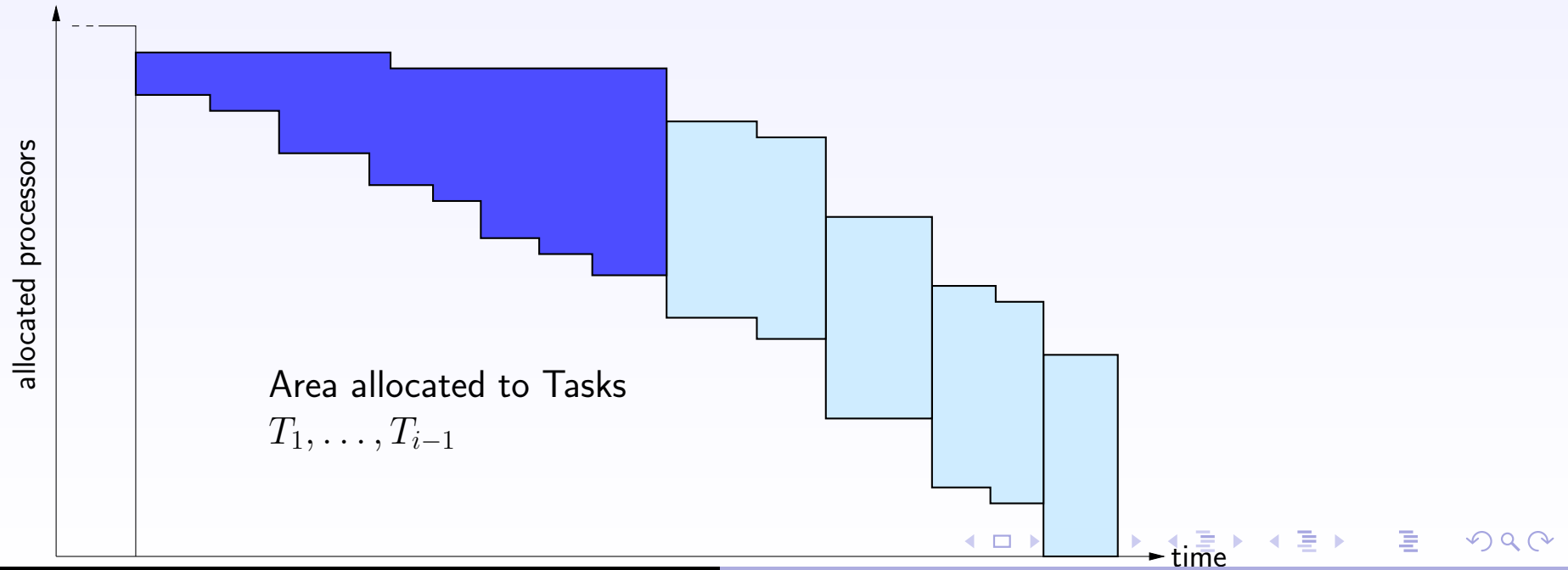
$$\begin{array}{lll} \text{MINIMIZE} & \sum w_i C_i & \\ & \forall i, \forall j \leqslant i, & x_{i,j} \leqslant (C_j - C_{j-1}) b_i \\ & \forall i, & \sum_{j \leqslant i} x_{i,j} = S_i \\ & \forall j, & \sum_i x_{i,j} \leqslant (C_j - C_{j-1}) B_0 \end{array}$$

- where $x_{i,j}, \forall j \leqslant i$ denotes the area allocated to $T_i$ in column $j$



# Normal Form (1)

- Any valid schedule with $C_1 \leqslant C_2 \leqslant C_n$
- can be turned into a normal form (with preserved $C_i$'s).
- WaterFilling Algorithm
  - Allocate $T_1$, then $T_2, \ldots, T_n$
  - maximize available resources after $T_i$'s allocation
  - by balancing as much as possible the height of Columns $1, \ldots, i$
  - Fill columns from left to right starting by the time step with lowest demand (non decreasing allocation)



# Normal Form : Correctness Proof (2)

## Lemma

*Let $H_j^i$ the quantity of non yet allocated resources to $T_1, \ldots, T_i$ in column $j$, then* WaterFilling *Algorithm maximizes*

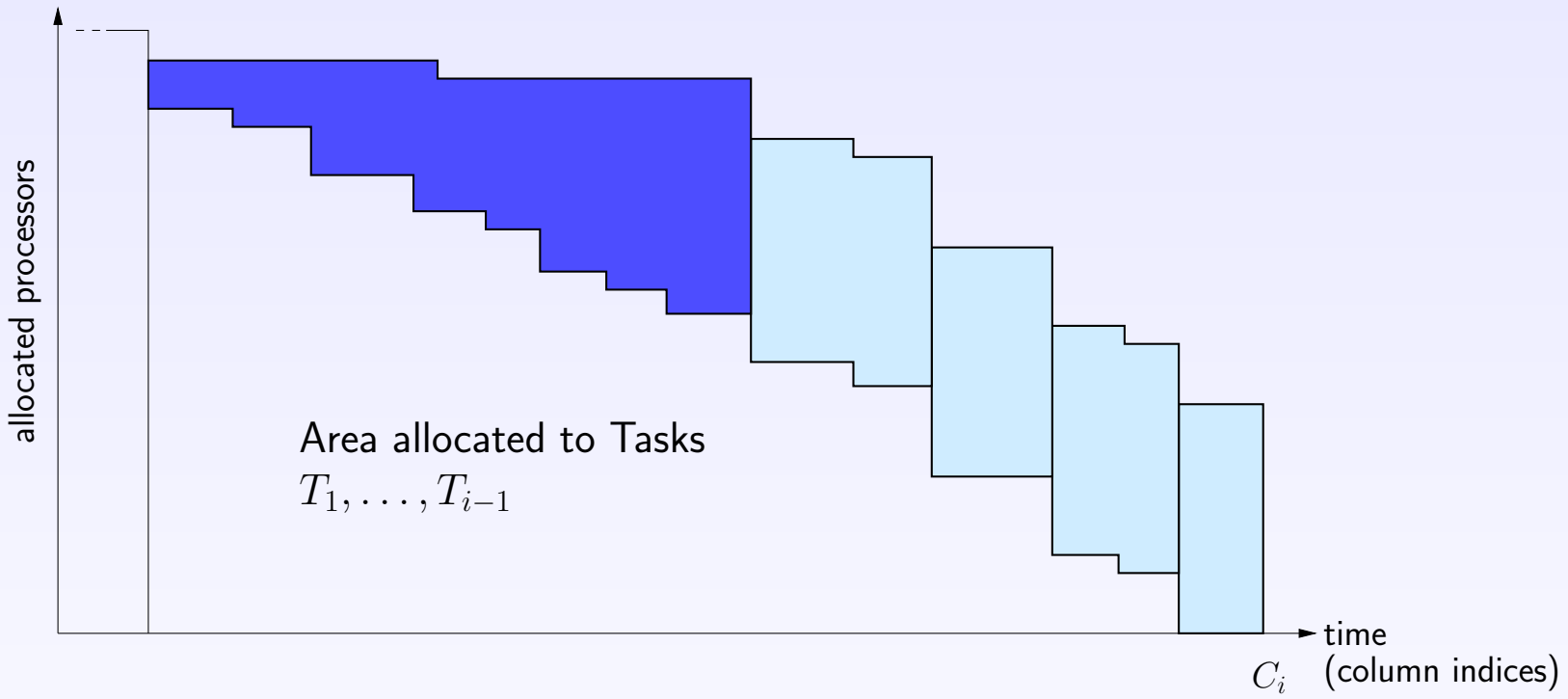
$$\sum_{k=1}^{i} \min(\delta, H_k^i) \times (C_k - C_{k-1}), \quad \forall \delta.$$

Whatever the resource limitation bound $\delta$ of the next task, it will be easier to allocate it if $T_1, \ldots, T_i$ have been allocated using WaterFilling Algorithm.

## Theorem

*Any valid solution can be put into the normal form generated by* WaterFilling *Algorithm.*

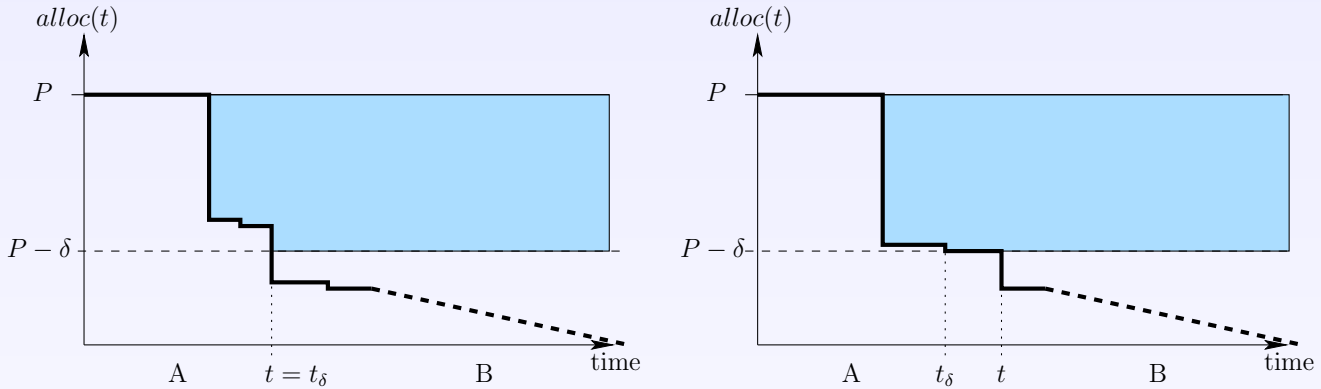
# Normal Form : Correctness Proof (3)




- either the maximal quantity of resources is allocated
- or the leftmost columns collapse into a single one
- possibly with a little step (integrity constraint)

# Normal Form : Correctness Proof (4)

$$\sum_{k=1}^{i} \min(\delta, H_k^i) \times (C_k - C_{k-1}) \text{ is maximal }, \quad \forall \delta.$$

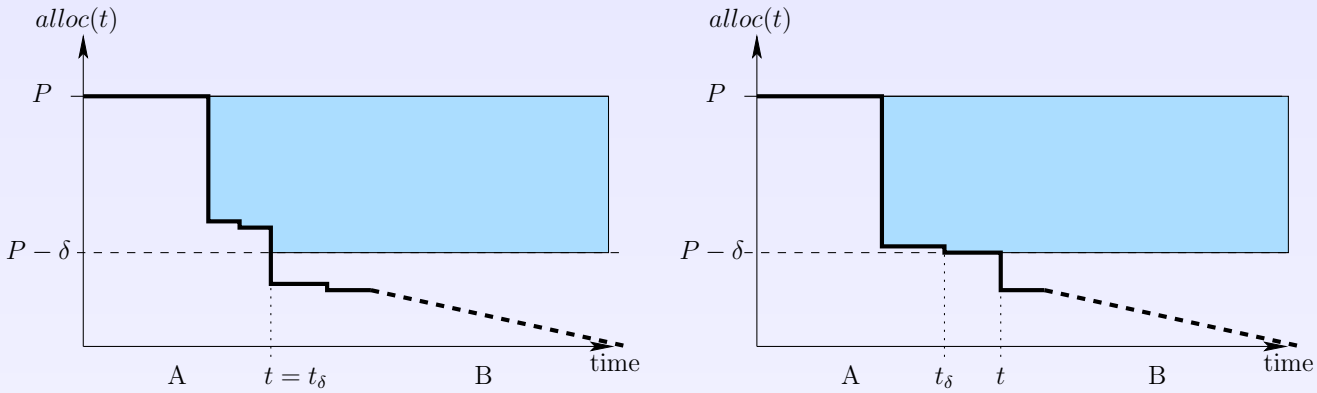



$t = C_a$,

$$\sum_{k=1}^{i} \min(\delta, H_k^i)(C_k - C_{k-1}) = \sum_{k=1}^{a} \min(\delta, H_k^i)(C_k - C_{k-1}) \quad (1)$$
$$+ \sum_{k=1}^{a} \min(\delta, H_k^i)(C_k - C_{k-1}) \quad (2)$$

- (2) clearly no allocation can give more than $\delta$ to the new task.
- (1) slightly more complicated

# Normal Form : Correctness Proof (5)

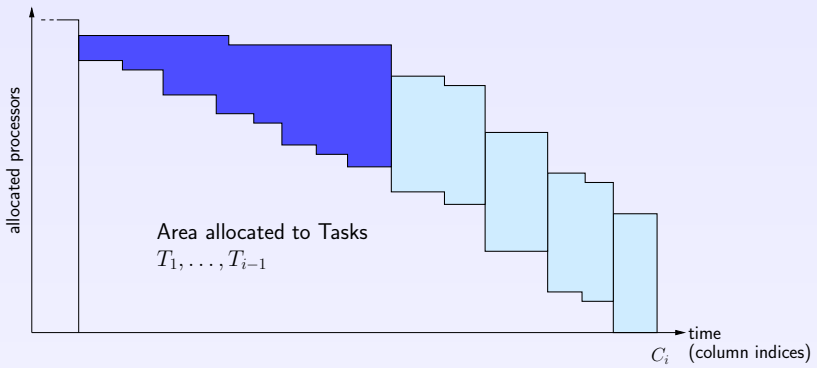



$$\sum_{k=1}^{i} \min(\delta, H_k^i)(C_k - C_{k-1}) = \sum_{k=1}^{a} \min(\delta, H_k^i)(C_k - C_{k-1}) \quad (1)$$
$$+ \sum_{k=1}^{a} \min(\delta, H_k^i)(C_k - C_{k-1}) \quad (2)$$

(1) contains terms corresponding to

- Tasks $T_i$, $i \leqslant a$ that end before $C_a$ (the area left to $C_a$ is $S_i$)
- Tasks $T_i$, $i > a$ and then
  - the area after $C_a$ is maximal ($\delta_i$ allocated)
  - since resources are allocated to $T_i$ both before and after $C_a$
  - and the maximal number of resources is allocated in all but the leftmost column for next tasks
  - and thus, the area before $C_a$ is minimal
- what achieves the proof !

# Normal Form : Number of Preemptions



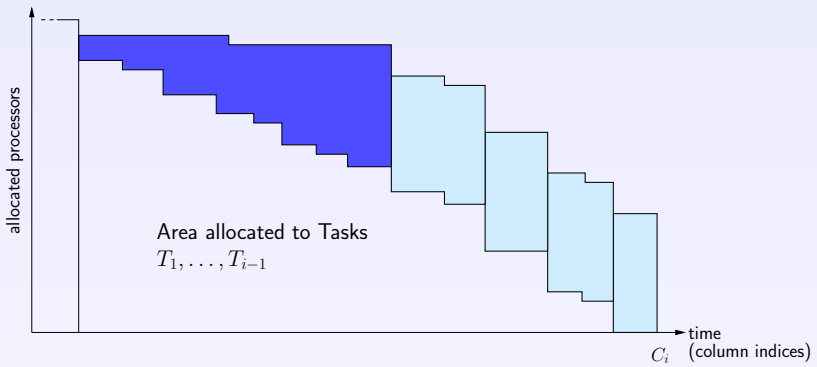

- $N_i$ overall number of changes in allocated resources for $T_1, \ldots, T_i$,
- $M_i$ overall number of changes in available resources after the allocation of tasks $T_1, \ldots, T_i$.

## Lemma

$$\forall i \geqslant 1, \quad N_{i+1} + M_{i+1} \leqslant N_i + M_i + 3.$$

and therefore, the average number of changes in resource allocation per task is bounded by 3.

# Normal Form : Number of Free Variables




- The allocation is completely defined by a tree
  - which tasks are covered by the leftmost column of a new task ?
- and thus $O(n)$ variables are enough to describe the solution instead of $O(n^2)$
- once the tree has been chosen

# Outline

# Conclusion and Open Problems

- Realistic communication model for Divisible Load Scheduling.
- DLS $\longleftrightarrow$ Malleable Tasks scheduling.
- Linear Cost Model :
  - NP Complete (proved, strong sense Open)
  - Very robust (scheduling does not matter (almost)... proved for $p = 2$, $p > 2$ Open)
- Fixed Cost Model
  - NP Complete (very close to Independent Malleable Tasks Scheduling)
  - Priority Based Schedules are enough (choose an ordering and use it to assign priorities) Open
  - Realistic Approximation Algorithm (at the moment 2-approx from the non-clairvoyant online case...) Open