Ordonnancement pour le parallélisme

Denis Trystram

trystram@imag.fr

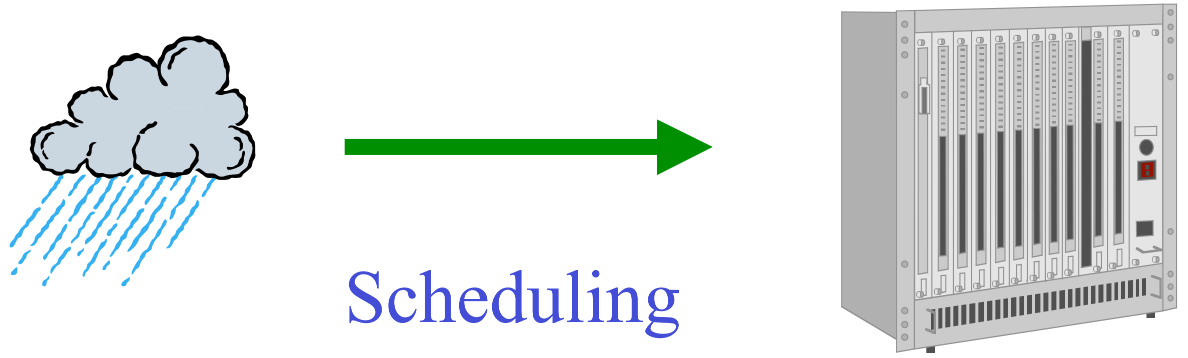
ID-IMAG

november 2003

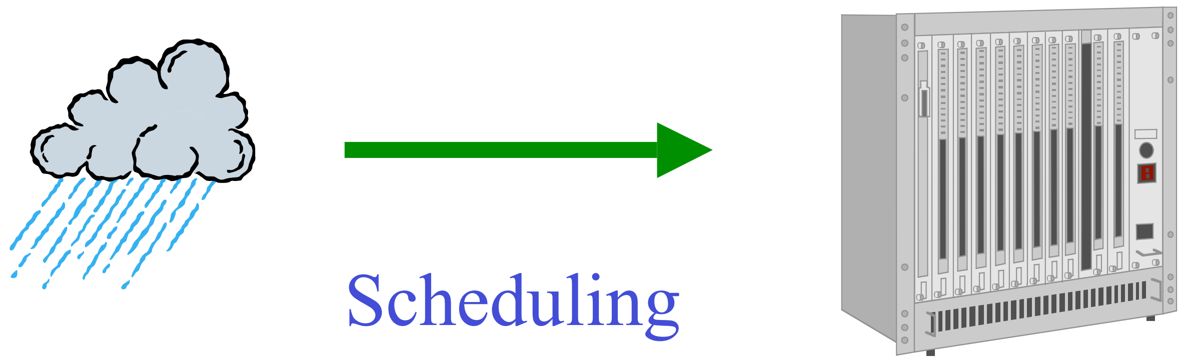
Contenu et Plan

- Context and Introduction
- Definitions and basic results
- Communication Delays
- Taking into account new characteristics
- Malleable Tasks
- On-line and multi-criteria optimization

Scheduling problem



Scheduling problem



Computational units are identifed and their relations are analyzed. We determine when and where they will be executed.

Target systems

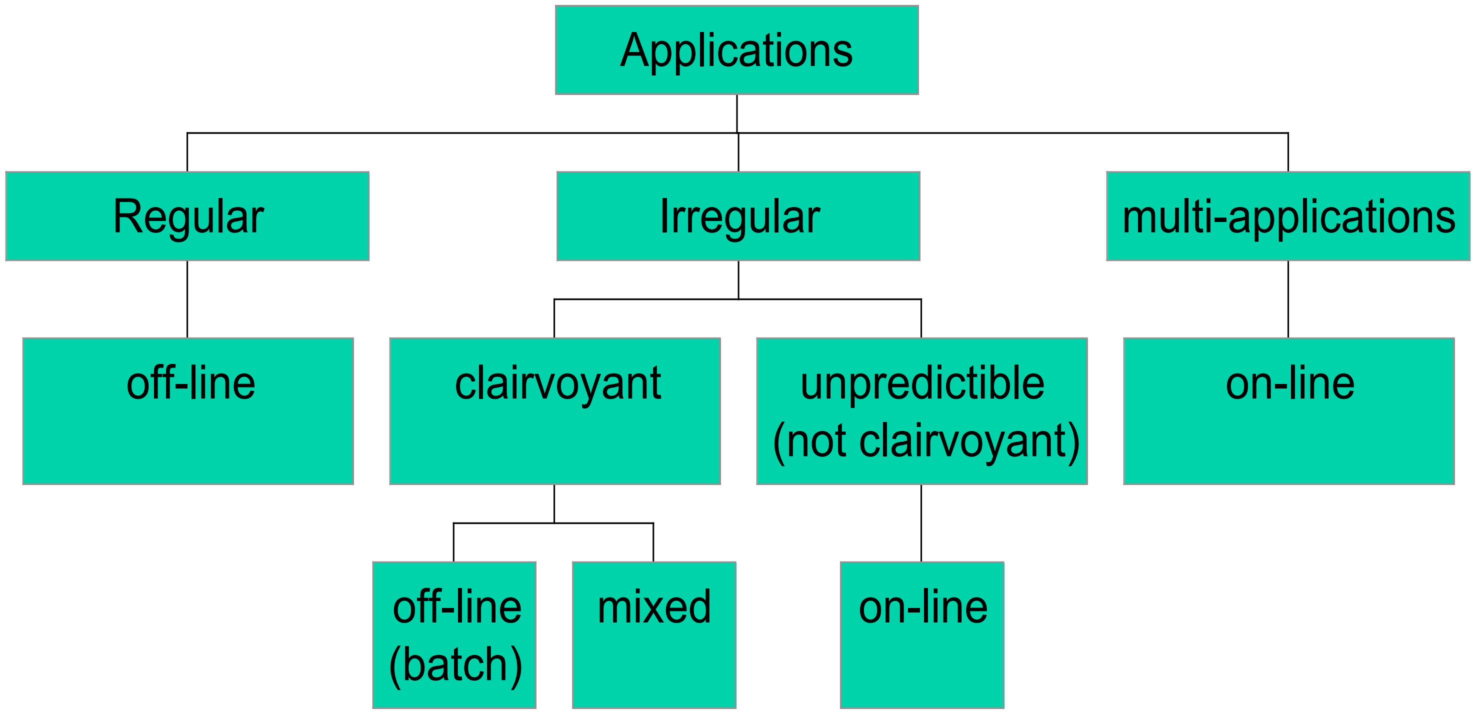
1980: shared-memory parallel systems and vector machines.

80-90: systèmes parallèles distribués très variés.

Années 90: systèmes distribués à gros grain à réseaux dynamiques.

Actuellement: grappes (hiérarchiques), calcul sur grilles hétérogènes et distantes.

Taxinomy of Applications



Precedence Task Graph

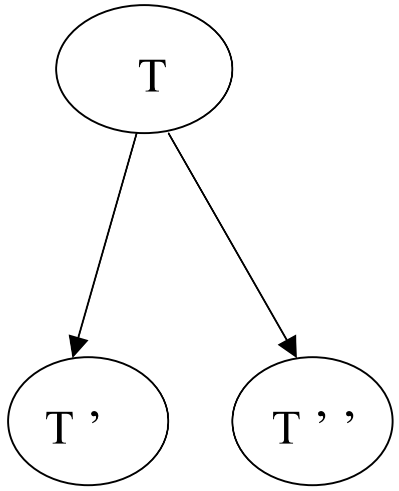
Let G=(V,E) be a weighted graph

$(i,j)\in E$ iff $i<<j$ (ordre partiel)

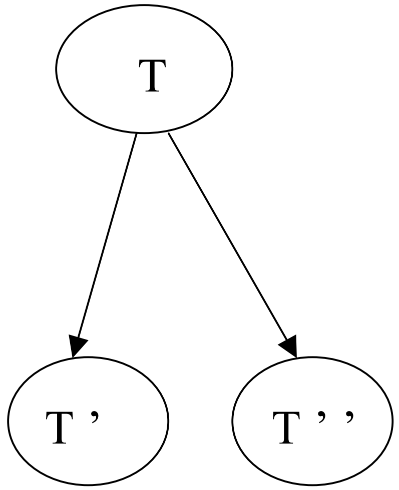
The vertices are weighted by the execution times.

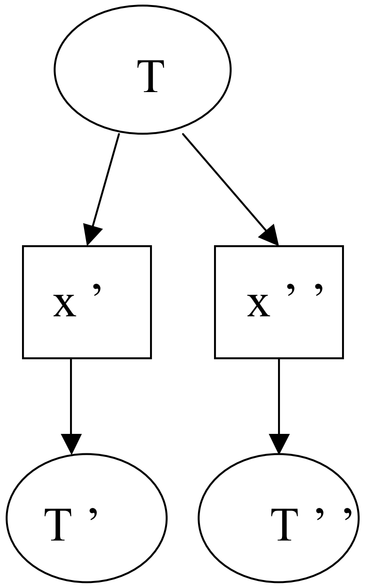
The arcs are weighted by the data to be transfered from a task to another.

Data-flow Graph



Data-flow Graph





bipartite graph

Example: computing C = AB by Strassen

Matrices A and B are partitionned by quadrant.

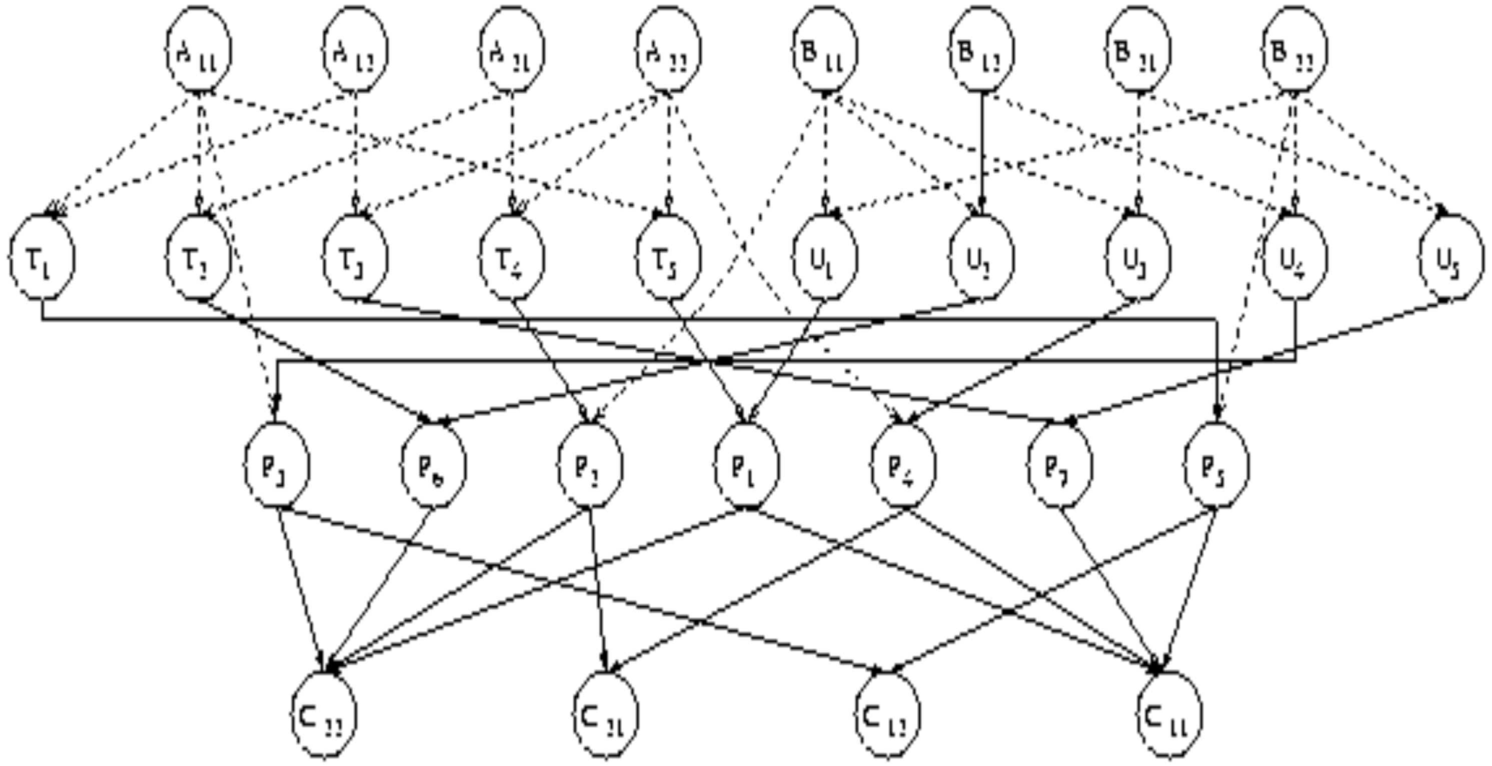
$$C_{12} = A_{11} * (B_{12} - B_{22}) + (A_{11} + A_{12}) * B_{22}$$

$$\begin{array}{|c|c|} \hline A_{11} & A_{12} \\ \hline A_{21} & A_{22} \\ \hline \end{array} \qquad \begin{array}{|c|c|} \hline B_{11} & B_{12} \\ \hline B_{21} & B_{22} \\ \hline \end{array}$$

Strassen: computing C = AB

$T_1 = A_{11} + A_{12}$; $T_2 = A_{21} - A_{11}$;
$T_3 = A_{12} - A_{22}$; $T_4 = A_{21} + A_{22}$;
$T_5 = A_{11} + A_{22}$;
$U_1 = B_{11} + B_{22}$; $U_2 = B_{11} + B_{12}$;
$U_3 = B_{21} - B_{11}$; $U_4 = B_{12} - B_{22}$;
$U_5 = B_{21} + B_{22}$;
$P_1 = T_5 * U_4$; $P_2 = T_4 * B_{11}$;
$P_3 = A_{11} * U_4$; $P_4 = A_{22} * U_3$;
$P_5 = T_1 * B_{22}$; $P_6 = T_2 * U_2$;
$P_7 = T_3 * U_5$;
$C_{11} = P_1 + P_4 - P_5 + P_7$; $C_{12} = P_3 + P_5$;
$C_{21} = P_2 + P_4$; $C_{22} = P_1 + P_3 - P_2 + P_6$;

Strassen's Task Graph



Formal Definition

The problem of scheduling graph G = (V,E) weighted by function t on m processors:

(without communication)

Determine the pair of functions (date,proc) subject to:

•respect of precedences

$$\forall (i,j) \in E : date(j) \geq date(i) + t(i, proc(i))$$

•objective: to minimize the makespan $C_{\max}$

3 fields notation

[Graham,Lenstra-Lageweg-Veltman1990]

b1|b2|b3

[Lenstra-Lageweg-Veltman,1990]

b1|b2|b3

- b1- resources and model

[Lenstra-Lageweg-Veltman,1990]

b1|b2|b3

- b1- resources and model
- b2 - graph and schedule

[Lenstra-Lageweg-Veltman,1990]

b1|b2|b3

- b1- resources and model
- b2 - graph and schedule
- b3 - objective

[Lenstra-Lageweg-Veltman,1990]

b1|b2|b3

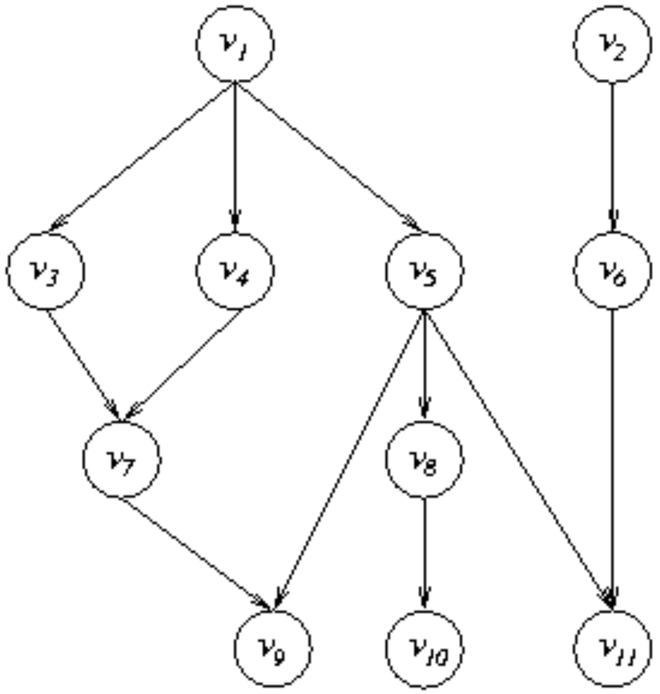
- b1- resources and model
- b2 - graph and schedule
- b3 - objective

Example: $P\infty \mid prec, pj \mid C_{\max}$

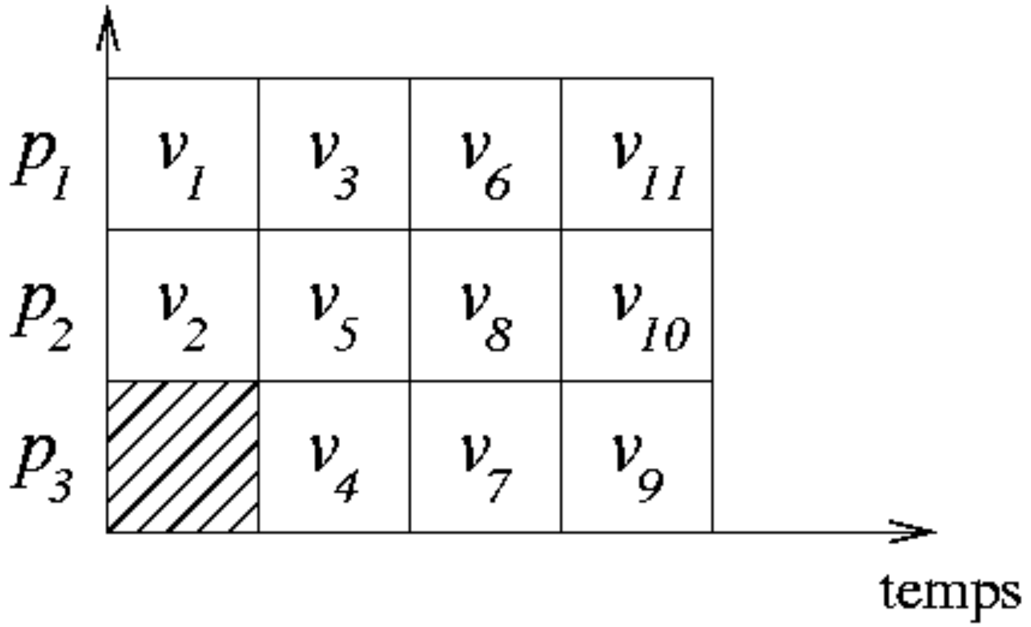
Parameters of a Problem

- b1- implicit, BSP, LogP, $P\infty$, P or Pm
- b2 - prec, trees, grids - dup, pmtn, pj, Cij
- b3 - $C_{\max}, \sum C_i$, overhead

Example



Scheduling without communication (m=3)



Theoretical Models

PRAM: modèles de référence pour la classification.

Shared-memory: ordonnancement pur, sans délais de communication. Grain fin et faiblement couplé.

Distributed-memory: prise en compte des communications (UET-UCT) explicites et modèles élargis (linéaires, LogP, etc..).

Grappes et Grilles: nouveaux paramètres.

Central Scheduling Problem

P | prec, pj | Cmax is NP-hard [Ulmann75]

Thus, we are looking for good heuristics.

•Competitive ratio r:

maximum over all instances of $\frac{\omega}{\omega^*}$

The schedule S is said ρ-competitive iff $r(\sigma) \leq \rho$

Some results

Pinf | prec, pj | Cmax is polynomial (longest path)

Pm | prec, pj=1 | Cmax is still open for m>2

P2 | prec, pj=1 | Cmax is polynomial

[Coffman-Graham72]

List scheduling

Principle: build first the list of ready tasks and execute them with any greedy policy (in any order when they are available).

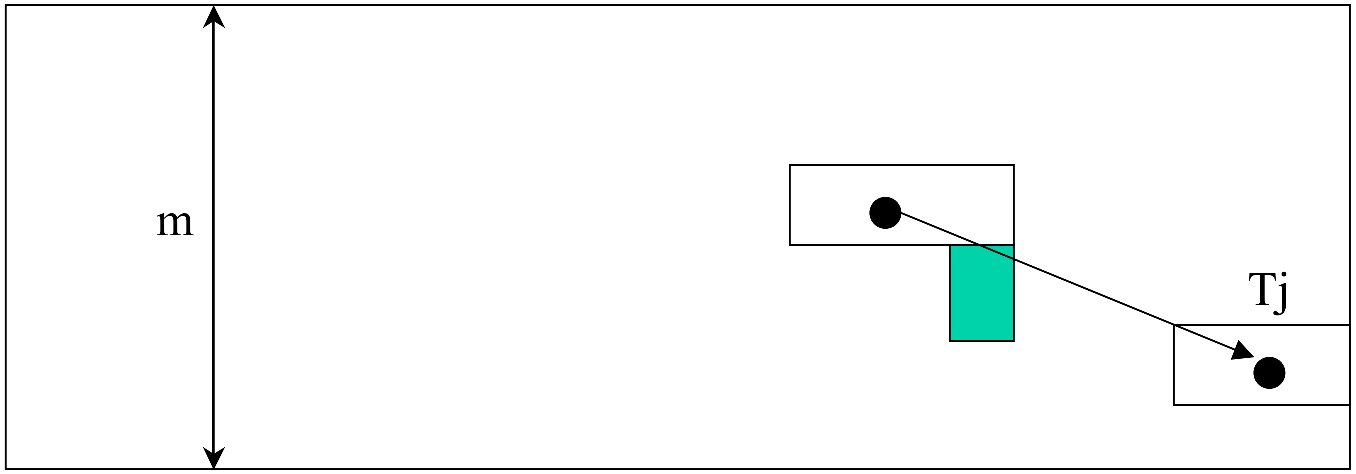
Pm | prec, pj | Cmax is 2-competitive

Analysis of list scheduling

We start from the end of the schedule:

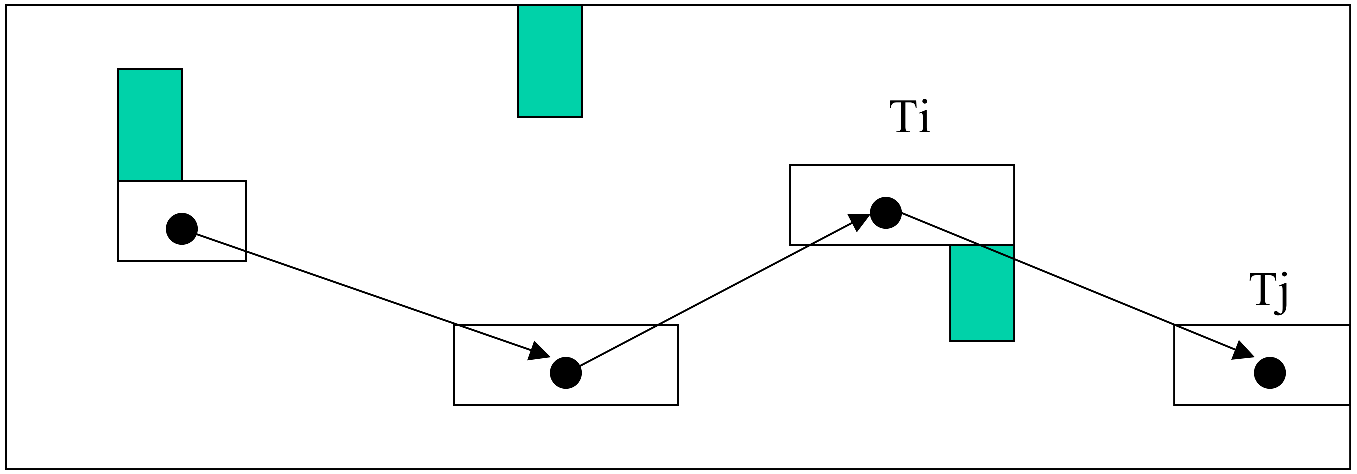
$\omega = \frac{W + idle}{m}$ where W is the total work

The idea of the proof is to bound the term idle



While there exist some time slots with idle periods:

there is one active task which is linked with Tj



We continue from Ti until it remains no idle time

Proof:

$$idle \le (m-1) l_{ch} \le (m-1) t_\infty$$

$$\frac{W}{m} \le \omega^*$$

$$\omega \le \omega^* + \frac{m-1}{m} t_\infty$$

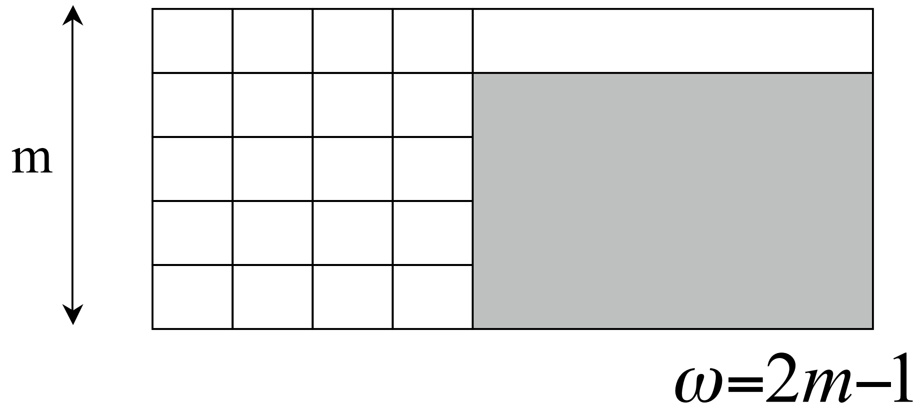
As the critical path is also a lower bound of the optimum:

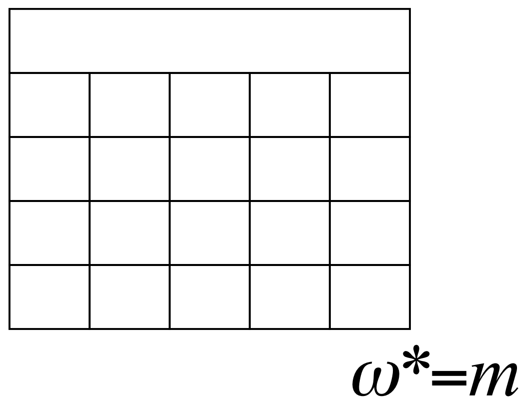
$$\omega \le \left(2 - \frac{1}{m}\right) \omega^*$$

Worst case

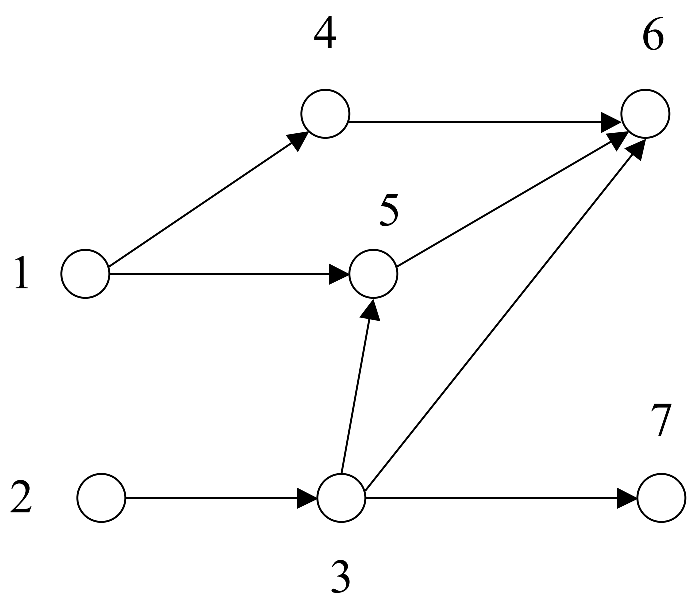
The bound is tight:

Consider (m-1)m UET tasks and 1 task of length m

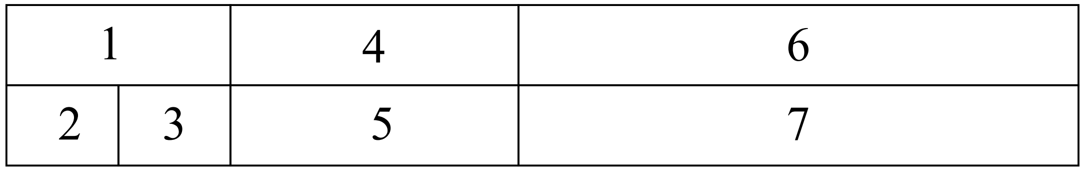




Anomalies [Graham]

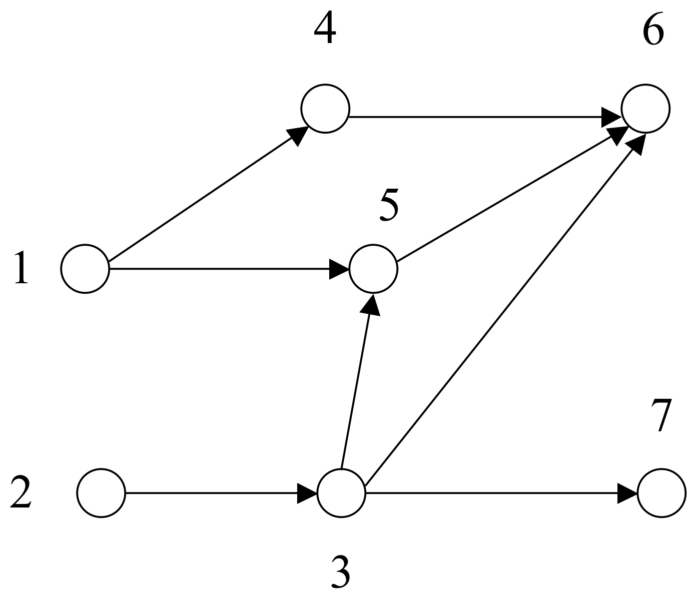


Weights: (4,2,2,5,5,10,10)



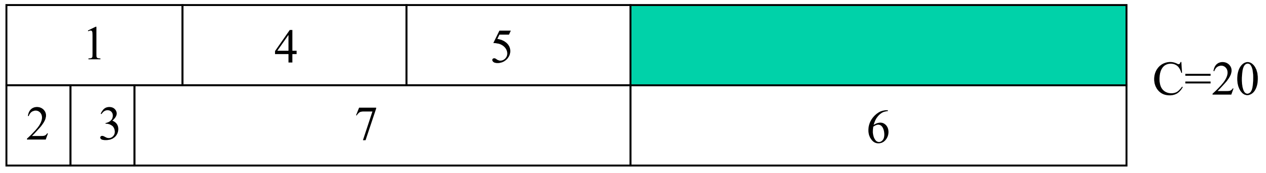
C=14

Anomalies [Graham]



All weights have one unit less:

(3,1,1,4,4,9,9)



Lower bounds

Basic tool:

Theorem of impossibility [Lenstra-Shmoys'95]

• given a scheduling problem and an integer c, if it is NP-complete to schedule this problem in less than c times, then there is no schedule with a competitive ratio lower than (c+1)/c.

Application

Proposition

The problem of deciding (for any UET graph) if there exists a valid schedule of length at most 3 is NP-complete.

Proof: by reduction from CLIQUE

Application

Proposition

The problem of deciding (for any UET graph) if there exists a valid schedule of length at most 3 is NP-complete.

Proof: by reduction from CLIQUE

Corollary: a lower bound for the competitive ratio of $Pm|prec,pj=1|C_{\max}$ is 4/3.

(finer) Upper Bound

Consider problem P |prec, pj=1 | Cmax

Proposition

There exists a (list-)algorithm whose performance guarantee is 2-2/m [Lam-Sethi,77] [Braschi-Trystram,94].

Proof adequate labeling of the tasks plus a priority based on the critical path.

Taking communications into account: the delay model

Introduced by [Rayward-Smith, 89]

•Total overlap of communications by local computations

•Possible duplication

•Simplified communications (unitary in the basic paper)

•No preemption allowed

Formal Definition

The problem of scheduling graph G = (V,E) weighted by function t on m processors:

(with communication)

Determine the pair of functions (date,proc) subject to:

•respect of precedences

$$\forall (i,j) \in E : date(j) \geq date(i) + t(i, proc(i)) + c(i,j)$$

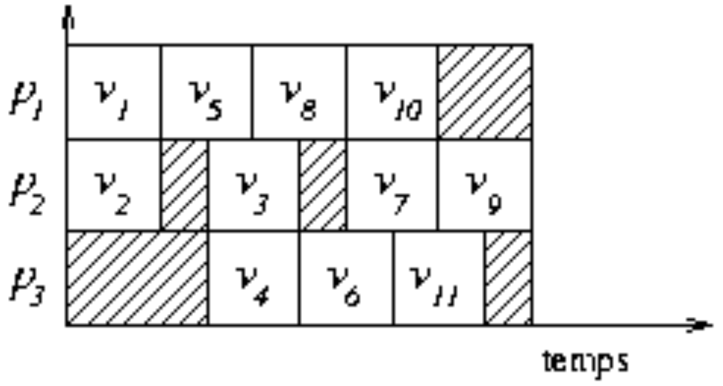
•objective: to minimize the makespan $C_{\max}$

Basic delay model

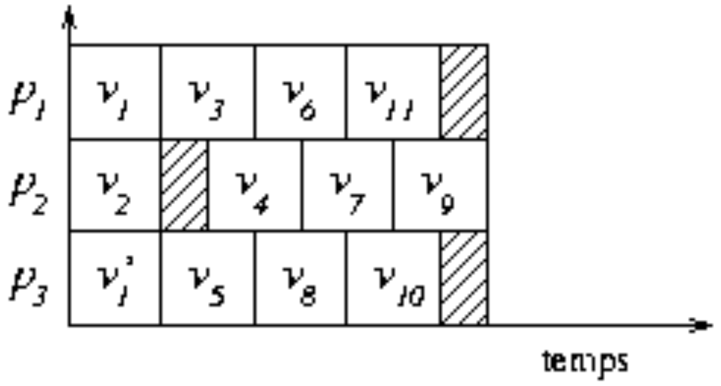
Comparing with no communication:

•Handling explicitly the communications is harder than the basic scheduling model

Scheduling with small delay with and without duplication

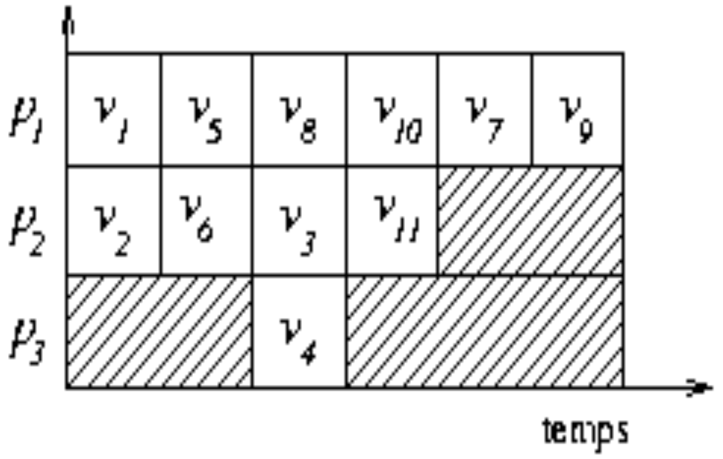


sans duplication

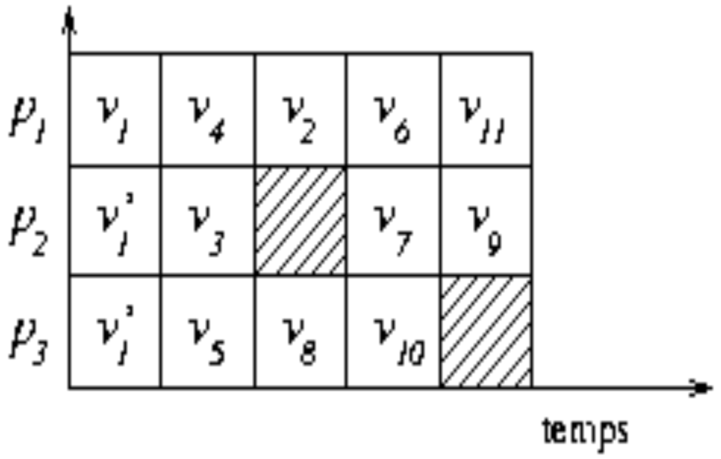


avec duplication

Scheduling with UCT delay with and without duplication



sans duplication



avec duplication

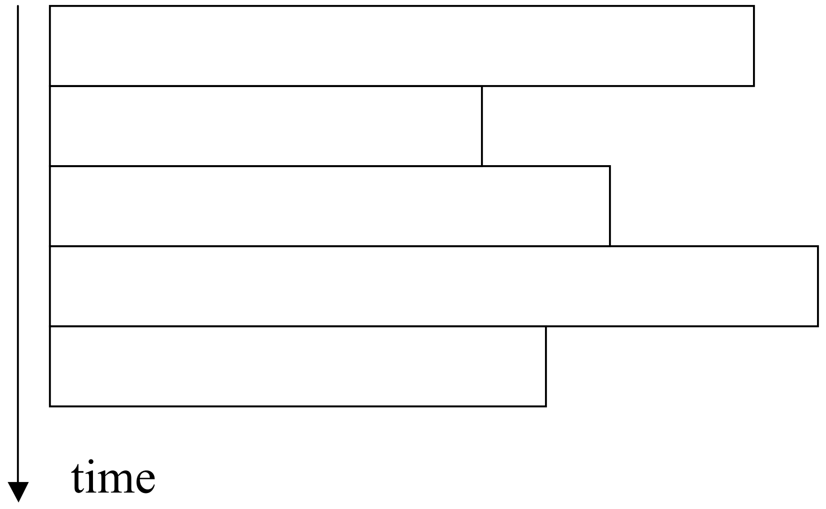
Brent’s Lemma

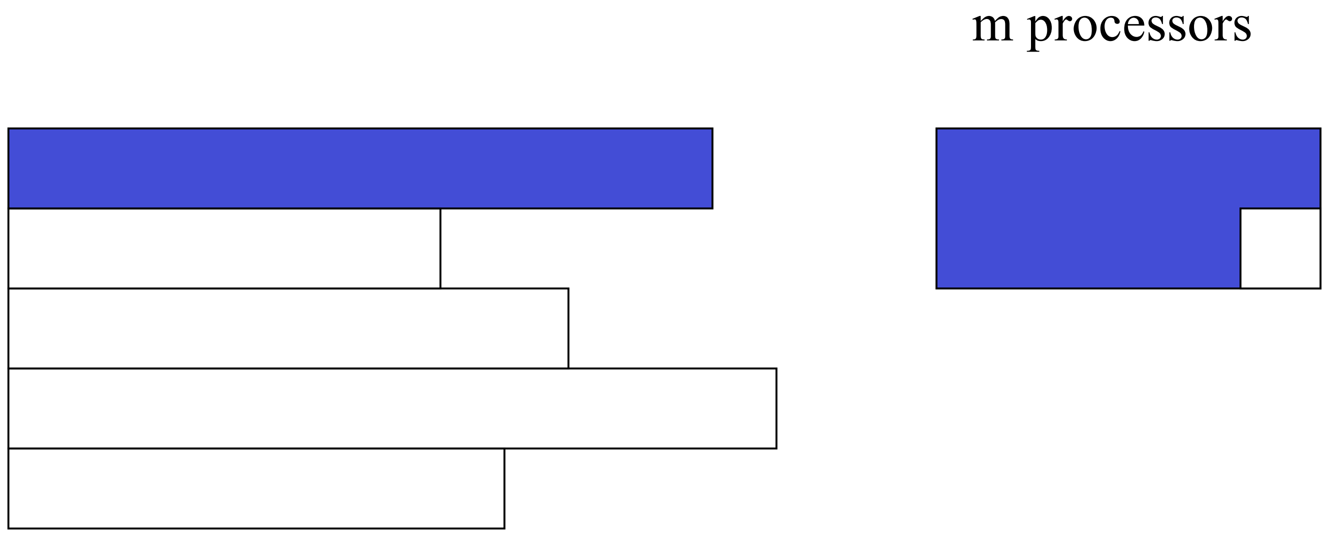
- Property:

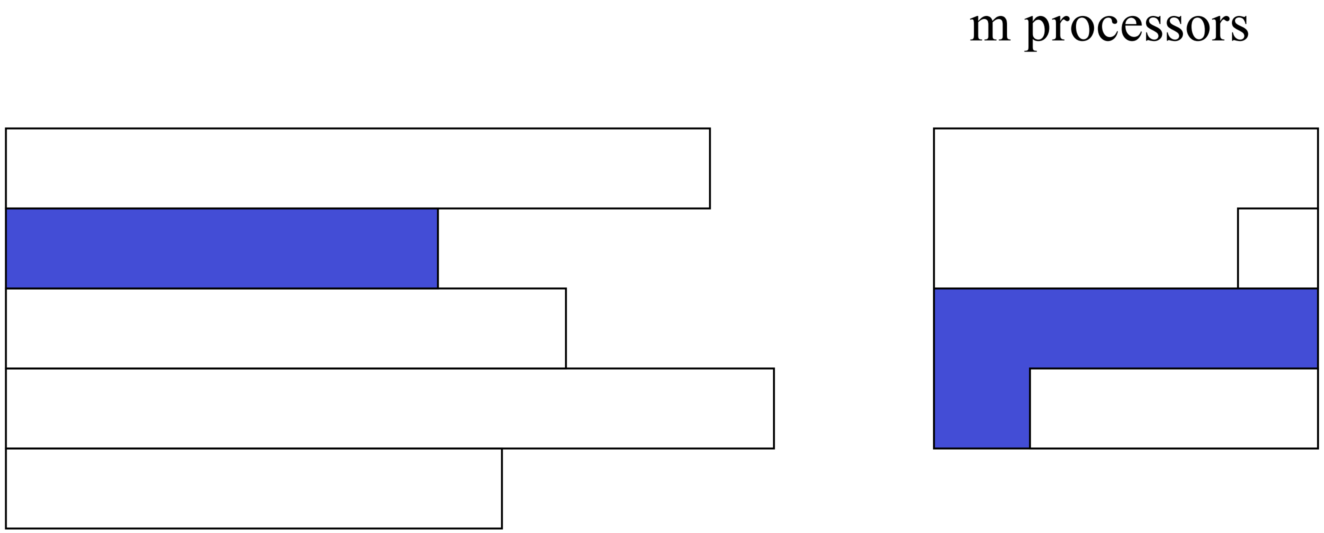
  let ρ be the competitive ratio of an algorithm with an unbounded number of processors. There exists an algorithm with performance ratio $\rho+1$ for an abritrary number of processors.

Principle

Gantt chart for m* processors




m processors


m processors

Proof

$\omega_m \le \omega^*_\infty + \omega_\infty$

(Similar to Graham's bound)

$\omega_\infty \le \rho \omega^*_\infty$

$\omega^*_\infty \le \omega^*_m$

Thus, $\omega_m \le (\rho+1)\omega^*_m$

Consequences: trivial Upper Bound

•As Pinf | prec, pj=1| Cmax is optimal (competitive ratio of 1), then:

P| prec, pj=1 | Cmax is 2-competitive.

•As Pinf | dup,prec, pj,cij| Cmax is 2-competitive, then:

P|dup, prec, pj, cij = 1 | Cmax is 3-competitive

List scheduling with communication delays

Trivial solution for UCT [Rayward-Smith]: insert systematically a communication at each step. 3-competitive algorithm.

Better solution for general graphs:

The principle is the same: just add a term proportional to the sum of the communications on the longest path [Hwang-Chow-Anger-Lee,89].

More sophisticated algorithms than list-algorithms

Formulation of $P|prec,pj=1,cij=1|Cmax$ as a ILP.

Xij are the decision variables

0 if task allot(i)=allot(j)

Solving as an ILP

Objective: minimize (C)

Constraints:

$$\forall i \in V, date(i) + 1 \leq C$$

$$\forall i \in V, date(i) \geq 0$$

$$\forall (i,j) \in E, date(i) + 1 + X_{i,j} \geq date(j)$$

$$\sum_{j} X_{i,j} \geq \mathrm{degree}(i)$$

$$X_{i,j} = 0,1$$

Solving as an ILP

Solve the LP with xij real numbers in [0,1]
and then, relax the solution:
xij < 0.5 are set to 0, the others are set to 1

Property:
this algorithm is 4/3-competitive.

Clustering Algorithms

Principle: unbounded number of processors. Starting from the smallest granularity, the tasks are gathered into subsets of tasks.

Property:

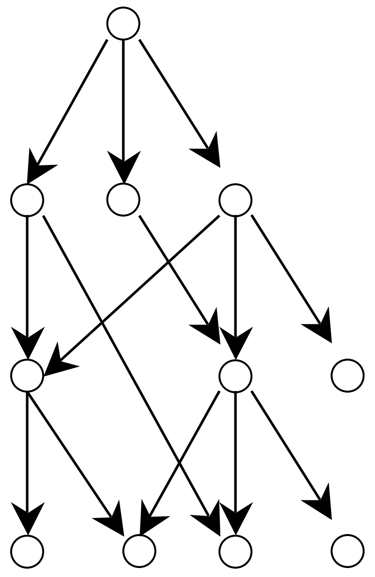
Critical path or maximum independent sets.

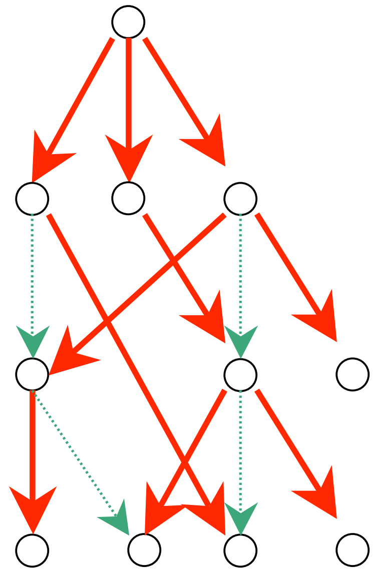
Influence of the duplication

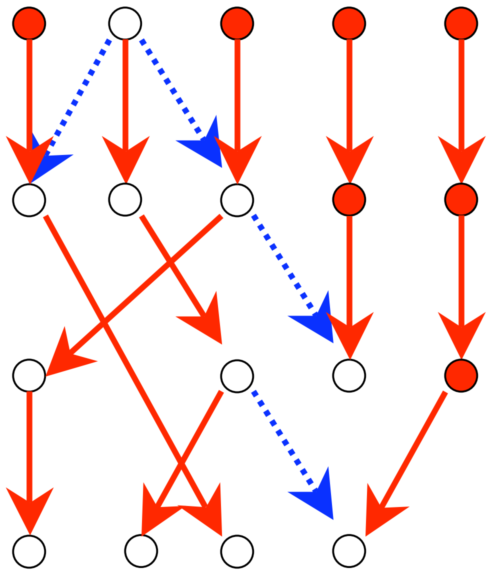
Pinf|prec,pj,cij<=1,dup|Cmax
is polynomial [Colin-Chretienne,90]

Idea: Find a spanning tree of minimum (local-) weights and schedule it by duplicating all the leaves.

Colin-Chrétienne







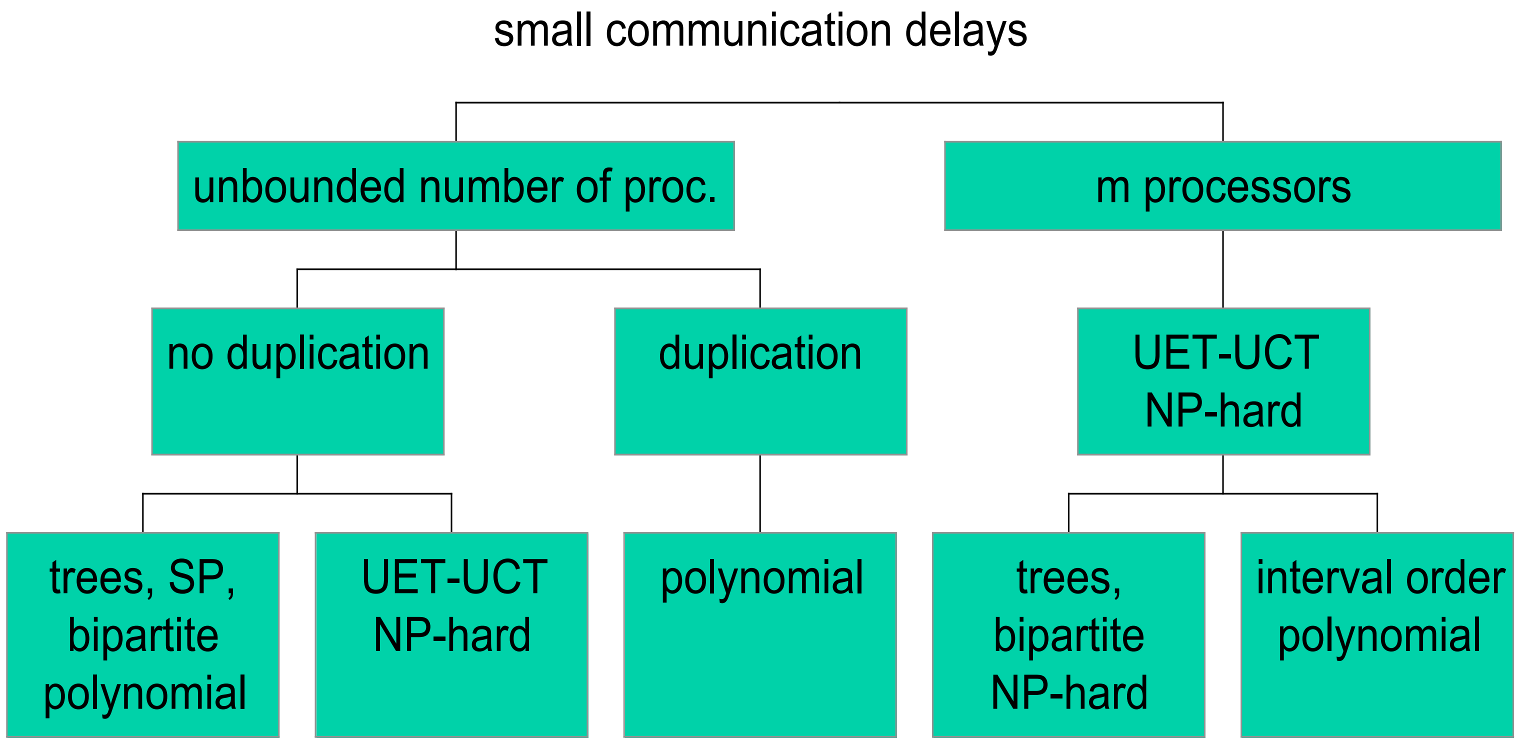
Duplication with a fixed number of processors

$P|prec,p_j=1,c_{ij}=1,dup|C_{max}$

is 2-competitive [Hanen-Munier,97]

Idea: by applying a list scheduling with duplication of parts of paths.

Synthesis


small communication delays
unbounded number of proc.
m processors
no duplication
duplication
UET-UCT NP-hard
trees, SP, bipartite polynomial
UET-UCT NP-hard
polynomial
trees, bipartite NP-hard
interval order polynomial

Scheduling with large delay

This problem is harder than with small communication delay

No competitive algorithm is known at this time

with a constant ratio (linear in the granularity factor)

Detailed result

Consider P | prec, pj=1, c>1 | Cmax

The best lower bound known at this time is 1+1/(g+3) [Bampis-Gianakos-Konig,98]

Practically, if g<<1 not interesting...

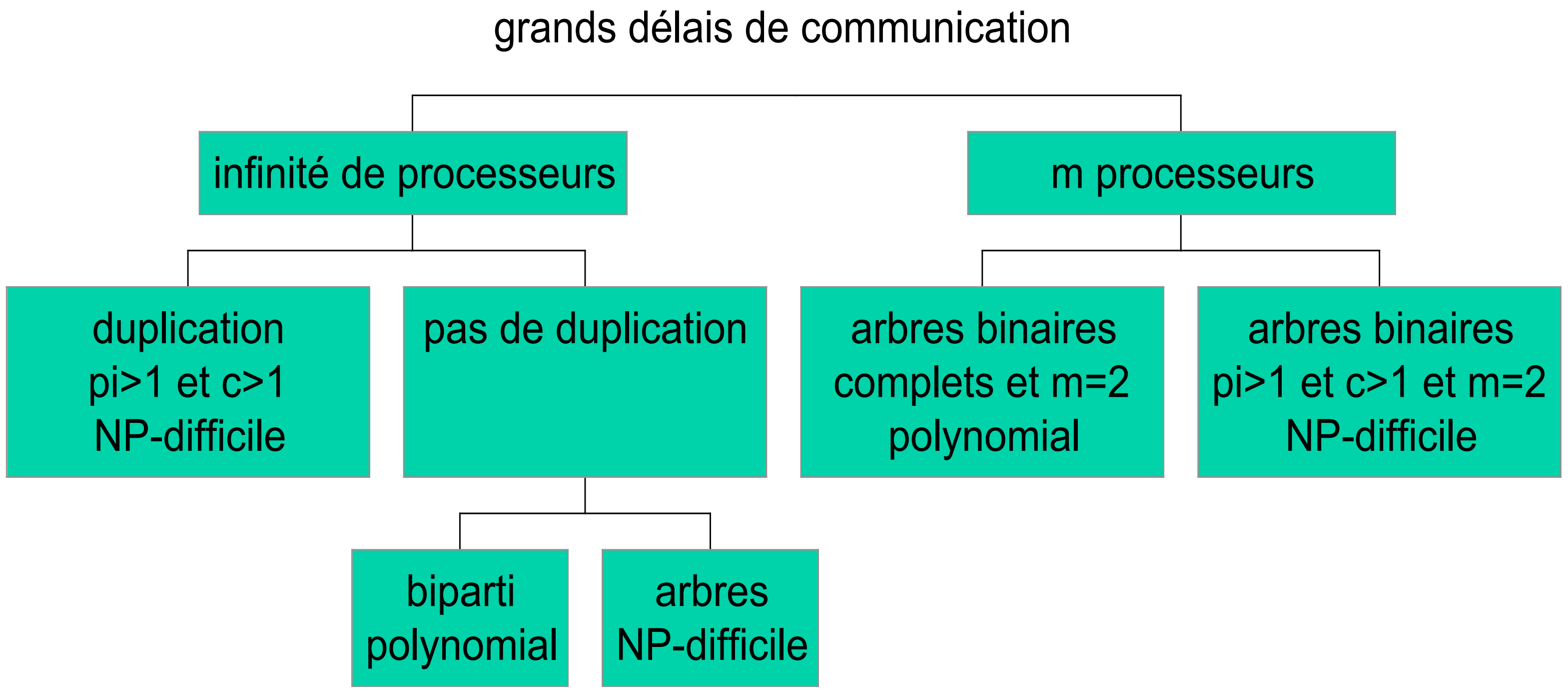
Large communication delays upper bound

Consider again P | prec, pj=1, c>1 | Cmax

The best upper bound known at this time is (c+2) [Bampis-Gianakos-Konig,97].

Another way to obtain this result is the trivial (list) algorithm which starts with no communication and systematically insert a communication between the computation steps...

Synthèse


grands délais de communication
infinité de processeurs
m processeurs
duplication pi>1 et c>1 NP-difficile
pas de duplication
arbres binaires complets et m=2 polynomial
arbres binaires pi>1 et c>1 et m=2 NP-difficile
biparti polynomial
arbres NP-difficile

Processeurs Uniformes (hétérogène)

Two natural extensions of the delay models are towards uniform (Q) and unrelated (R) processors.

NP-hard for very simple problems

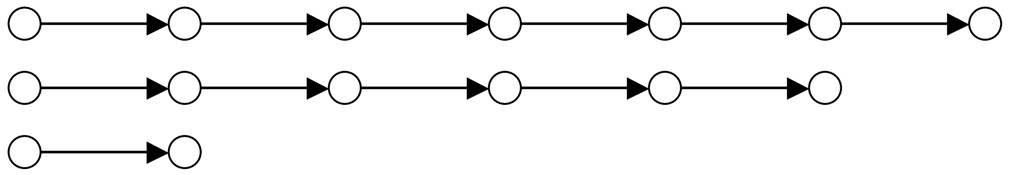
NP-hard for 1 machine plus a set of (m-1) identical machines

Scheduling independent chains

Qm|chains,pj=1,c=1|Cmax is strongly NP-hard while

Pm|chains,pj=1,c=1|Cmax is linear.

Example: scheduling chains on 2 processors (v1=1,v2=2).



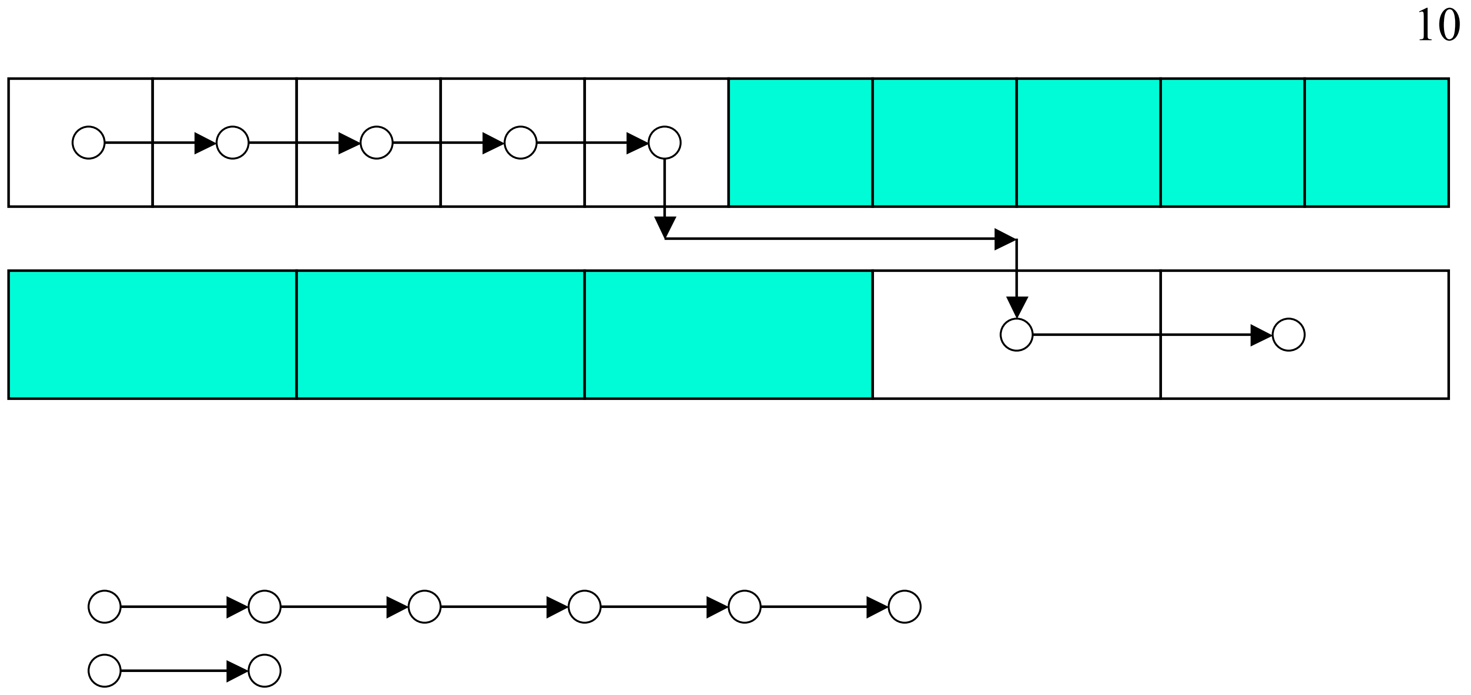
n1=7
n2=6
n3=2
Total n=15

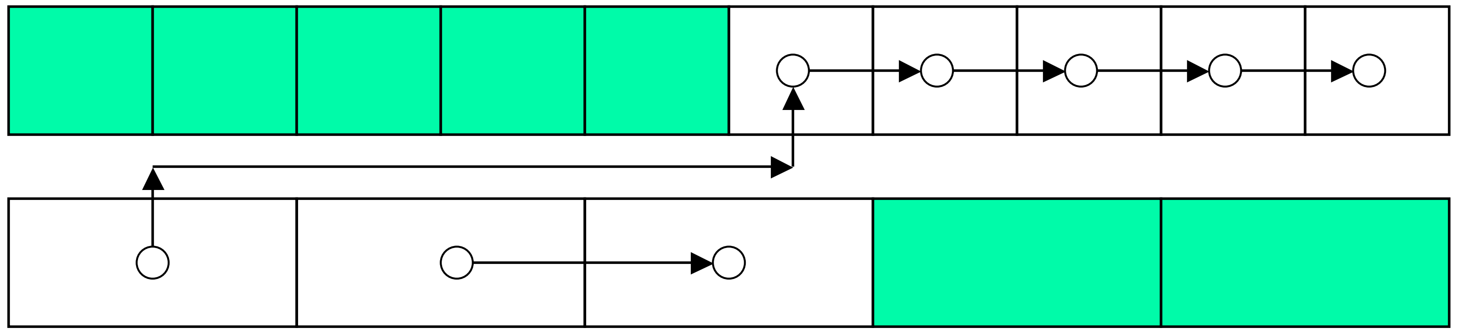
$$\omega \geq \max\left(\frac{v2(n1+n2)}{v1+v2}, n1\right) = 10$$

Idea: compute the maximum number of tasks to allocate to the slowest processor.

$\alpha v_2 + n_1 - \alpha < \omega^*$

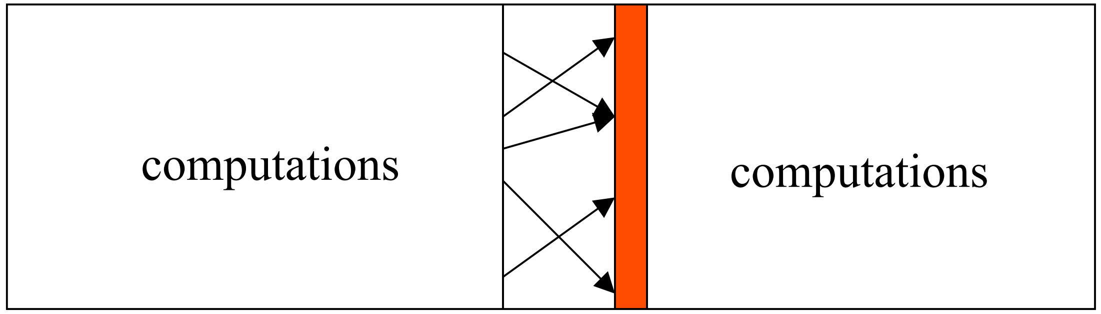
$\alpha = 2$





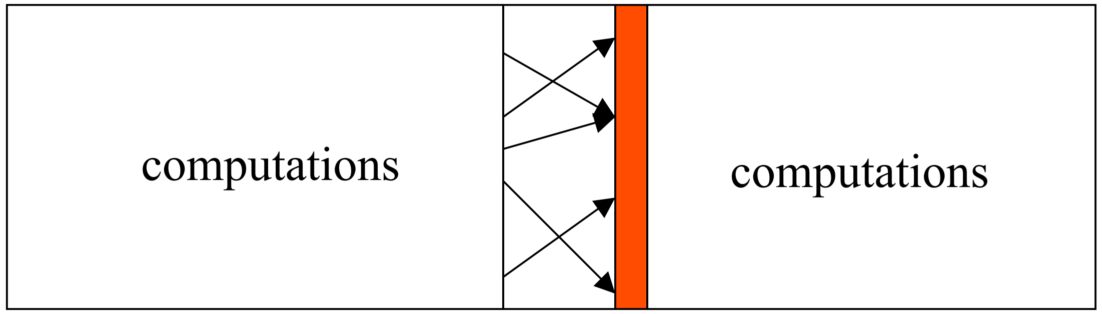
Alternative models: BSP

BSP is a programming paradigm [Valiant,90] whose principle is a series of independent steps of computations and communication-synchronization.



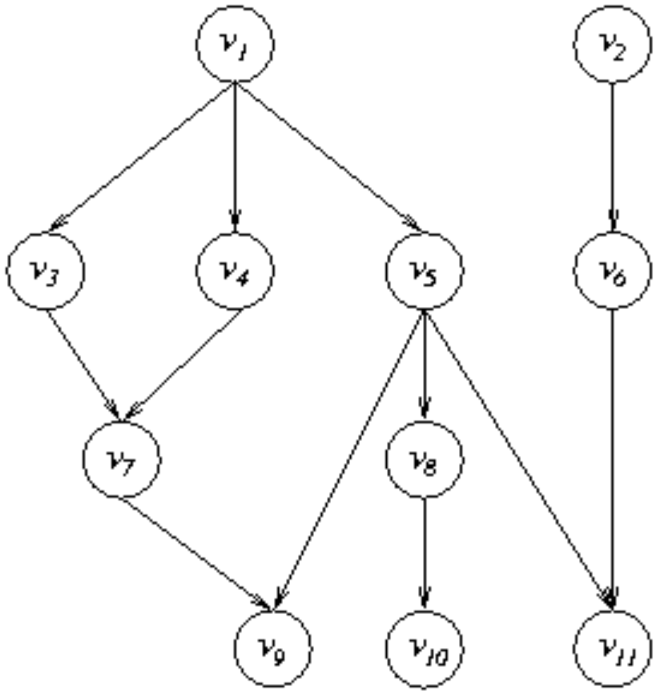
Alternative models: BSP

BSP is a programming paradigm [Valiant,90] whose principle is a series of independent steps of computations and communication-synchronization.

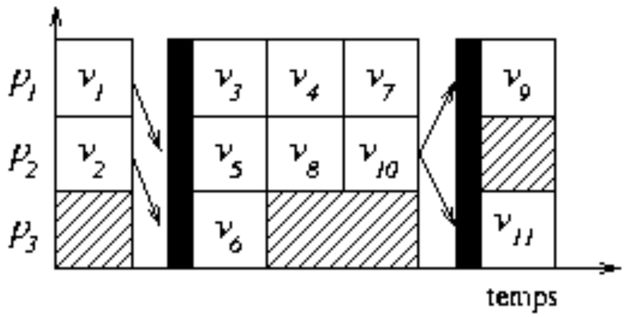


Scheduling under BSP is finding a tradeoff between load-balancing and number of CS

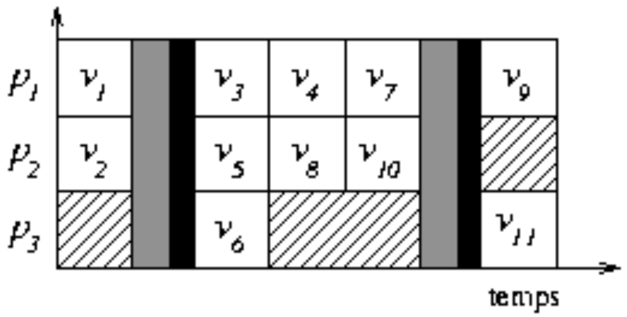
Coming back to the example



Scheduling in BSP



communications explicites



communication implicites

Parameters of BSP

•Latency (minimum time between communications)

•computing an h-relation (hg+s)

•based on a cost function

Complexity under BSP

- Simple problems under the delay model become hard under BSP
- However, it seems possible to design good competitive algorithms (for instance for scheduling independent chains).

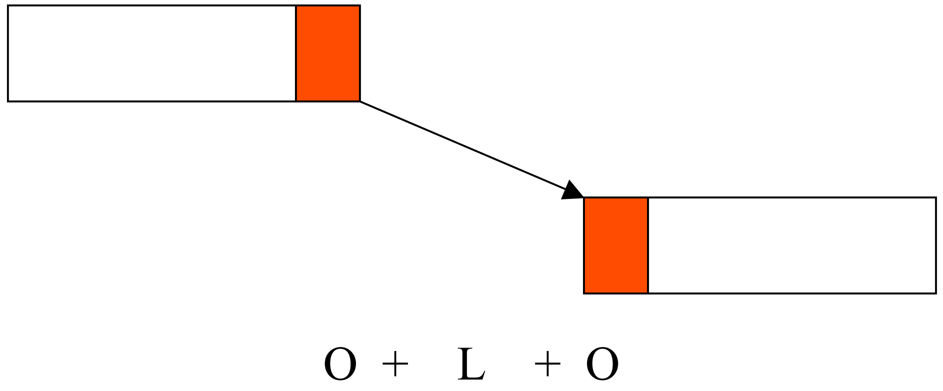
Alternative models: LogP

Need of computational models closer to the actual parallel systems [Culler et al.]: 4 parameters.

- L latency
- o overhead
- g gap
- P number of processors

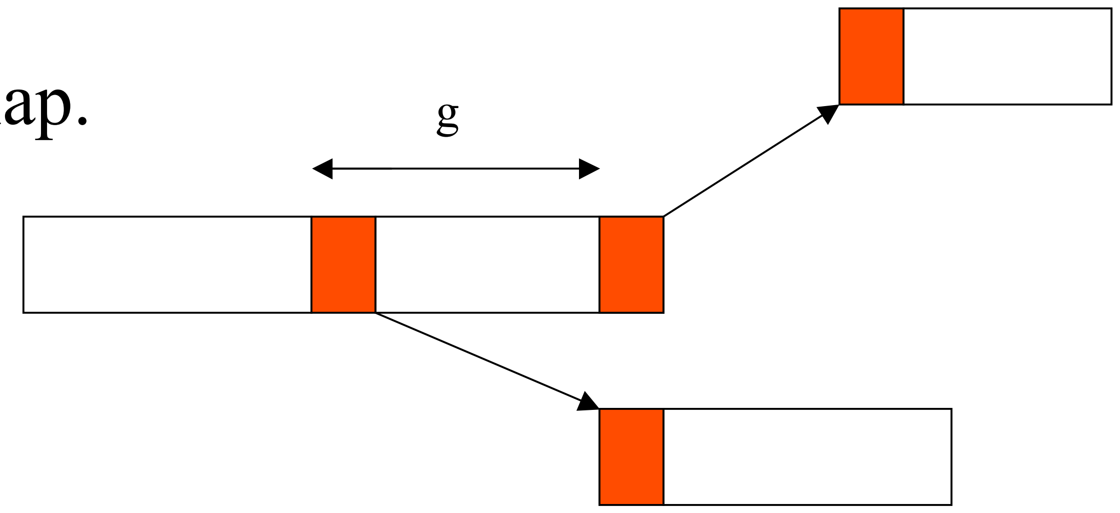
Alternative models: LogP

No overlap.



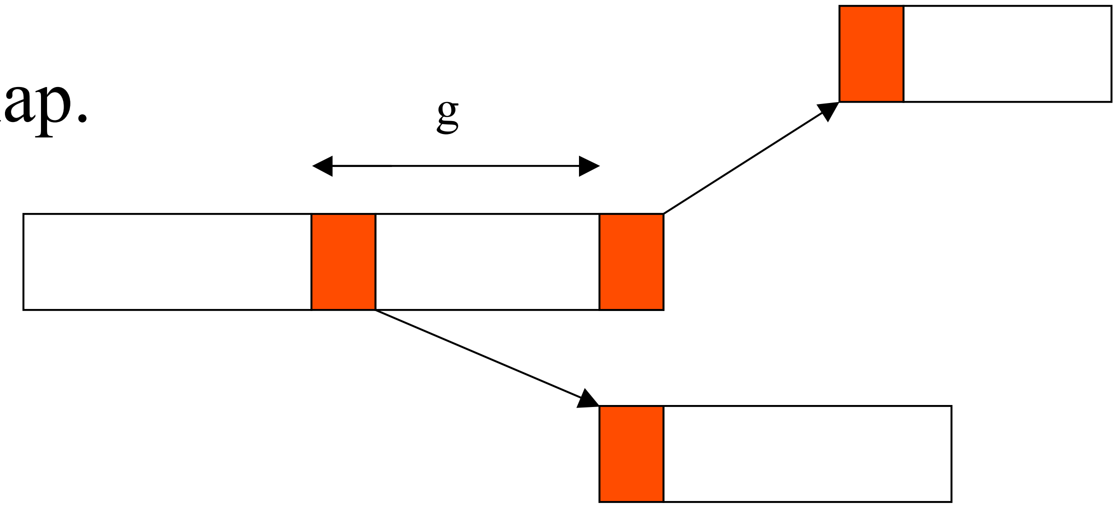
Alternative models: LogP

No overlap.



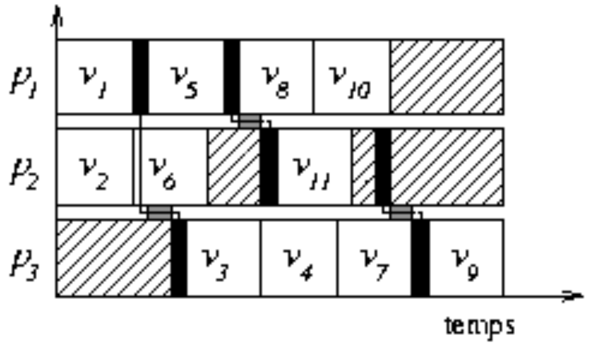
Alternative models: LogP

No overlap.

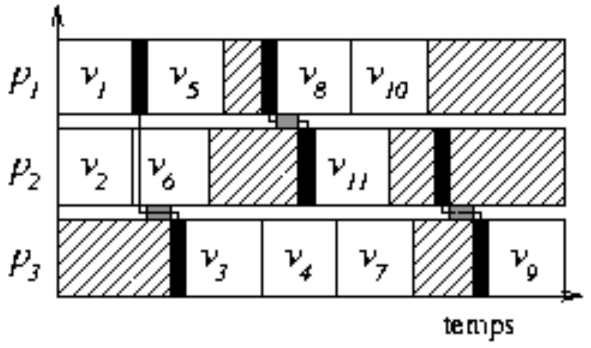


The delay model is a LogP-system where o=g=0

Scheduling the previous example in LogP



$g<1$



$g=1,5$

Complexity of LogP

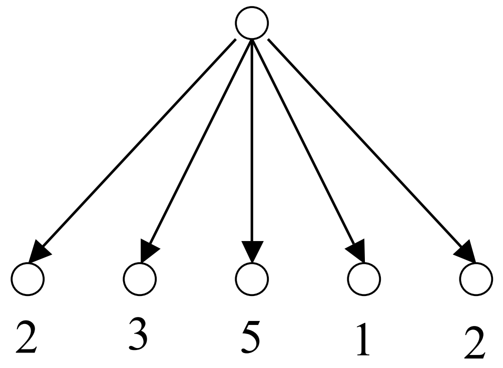
Of course, LogP seems (is?) harder.

It is true for

(LogP)Pinf | Fork,pj | Cmax and

(LogP)P=2 | Fork,pj | Cmax

Scheduling a fork graph under LogP



LogP is harder.

Alternative models : Malleable Tasks

Malleable Tasks

Communications are implicit

Natural link with applications:

• Partitioning the graph into routines.

• Parallel routines that can be analyzed easily (prediction of performances, using for instance PRAM algorithms or library routines).

Malleable Tasks

Informal definition:

A malleable task (MT) is a computational unitthat can itself be executed in parallel on an arbitrary number of processors.

History of MT

[Du-Leung] SIAM J.Discrete Maths 89

[Turek] SPAA’92

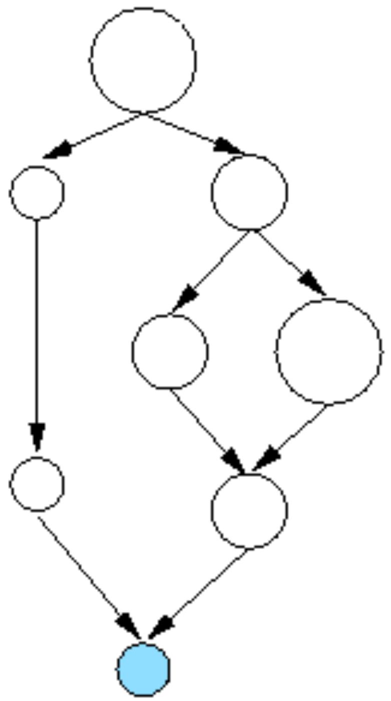
Related to:

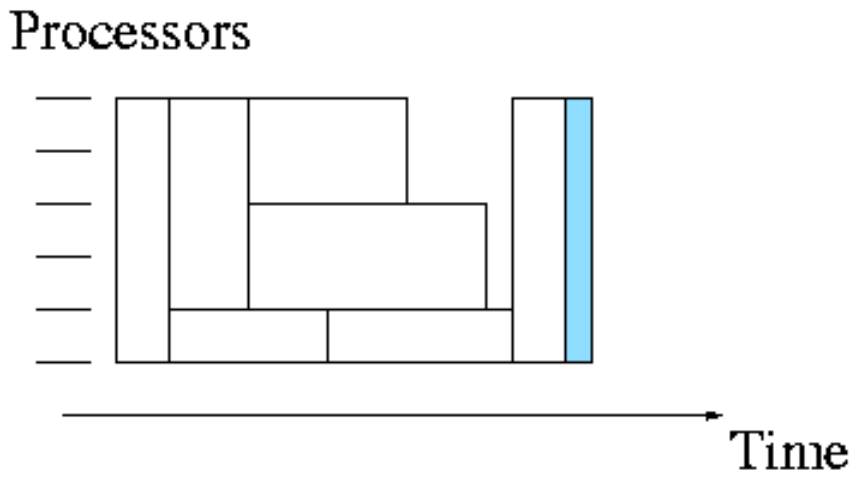
•Resource constraint scheduling [Graham75].

•Multiprocessor Tasks [Lloyd81], recent survey of [Drozdowski] in EJOR 86.

•Divisible Tasks [Srinivasa Prasanna-Musicus96] [Robertazzi]

Exemple


MT Graph



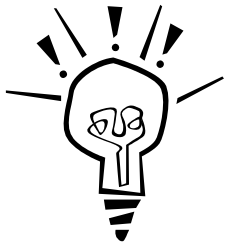
MT Scheduling

Advantage of MT

The granularity is large, thus, it allows to neglect communications between MT or at least to consider the SCT assumption…

The performance analysis of each MT can give a rather good execution time estimation

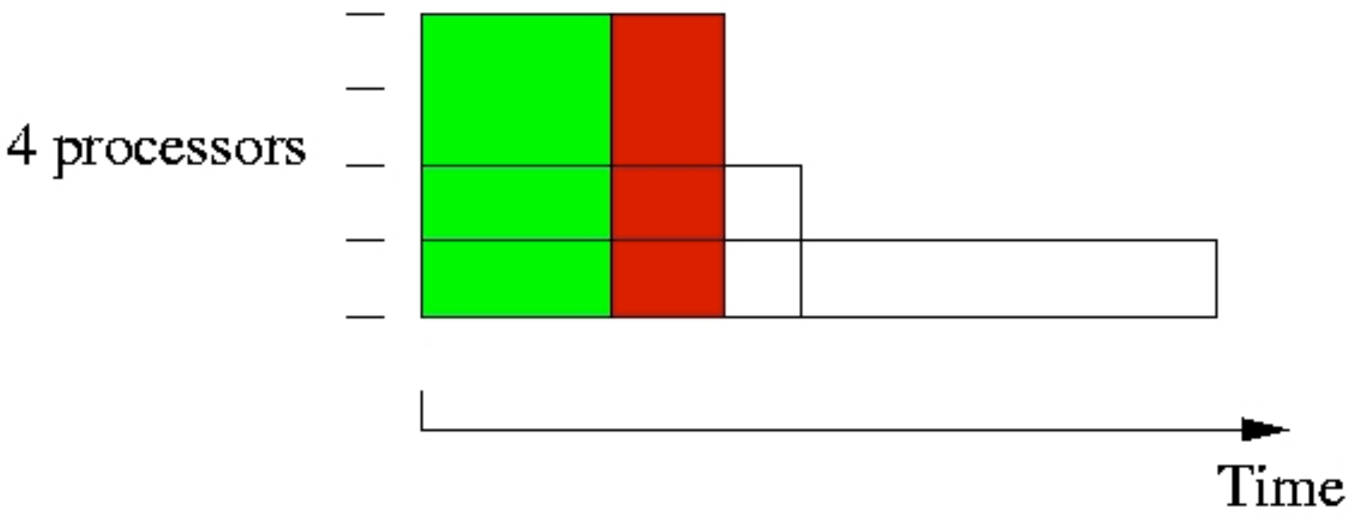
Taking into account the communications



We introduce a penalty factor for representing the global overhead (communications plus synchronizations plus sequential character).

Penalty

Le temps d'exécution parallèle décroit avec le nombre de processeurs et la pénalité augmente.



More Formally

Definition of Inefficiency factor of task T on i processors whose execution time is exec(T,i):

$$\mu(T,i) = \frac{exec(T,i)i}{exec(T,1)}$$

Expected Properties:

$$\mu(.,i) \uparrow$$

$$\frac{\mu(.,i)}{i} \downarrow$$

Formal definition

Scheduling of the MT-graph G = (V,E) on m processors:

Find two functions (date,allot) suject to:

- resource constraint

$$\forall \tau: \sum_{i \in slot\tau} allot(i) \leq m$$

- respect of precedences

$$\forall (i,j) \in E:$$

$$date(j) \geq date(i) + t(i, allot(i)) + C_{i,j}$$

•objective: minimizing the makespan $C_{\max}$

Scheduling MT

Principle in two successive phases (allocation et packing)

Two types of algorithms:

- simple allotment – complex multi-processor scheduling
- sophisticated allotment – simple multi-processor scheduling

General approach

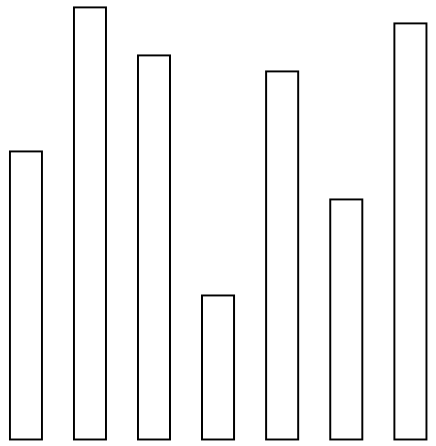
To focus on the allotment

- L’allocation est sophistiquée, réalisée par un algorithme de Programmation Mathématique (type Sac-à-dos).
- Utilisation d’un algorithme d’ordonnancement « simple » (liste).

K dual Approximation

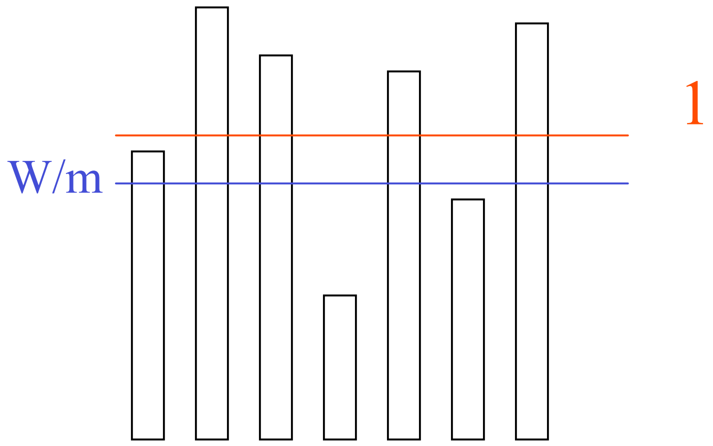
- On fixe une valeur de l’objectif d (entier).
- On applique un algorithme, si la garantie obtenue est plus mauvaise que kd, on recommence (dichotomie).

Independent MT
as a case study

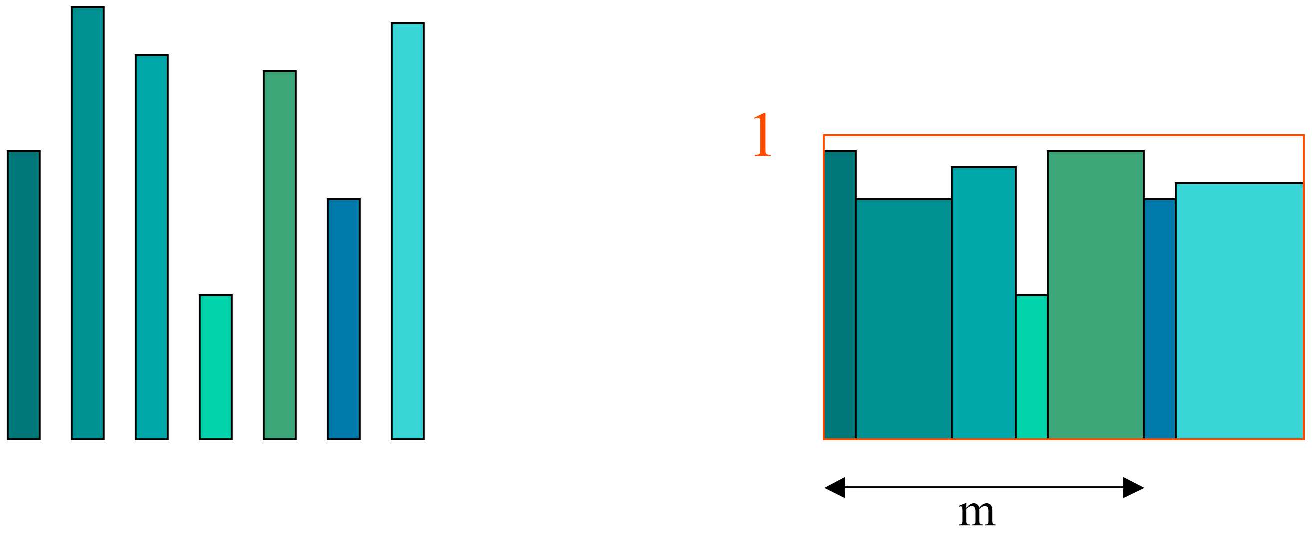


Let us consider 7 MT to be scheduled on m=10 processors.

Canonical Allotment



Canonical Allotment



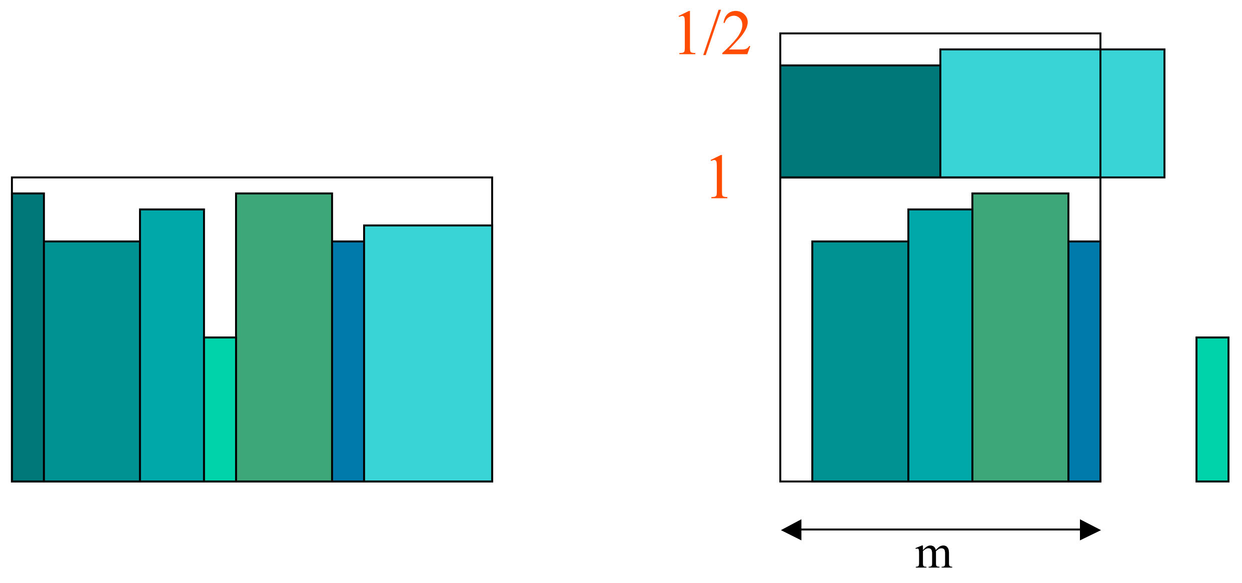
Maximal number of processors needed for executing the tasks in time lower than 1.

2-shelves scheduling

Idea: to analyze the structure of the optimum where the tasks are either greater than ½ or not.

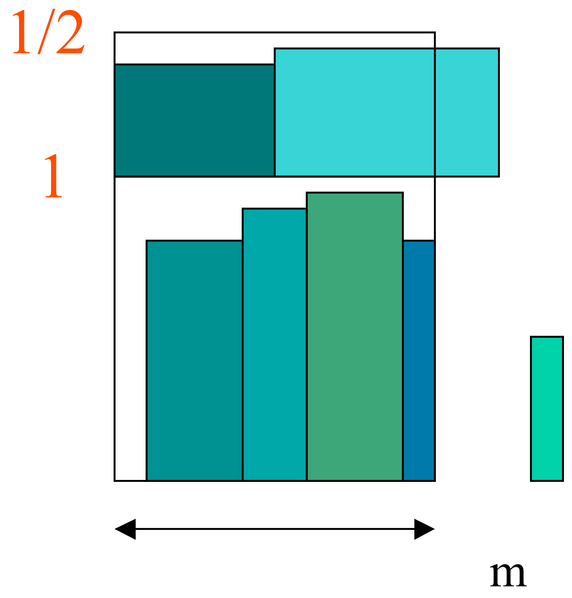
Thus, we will try to fill two shelves with these tasks.

How to select the tasks?



Knapsack: min(W*) s.c. # processeurs de S1<m

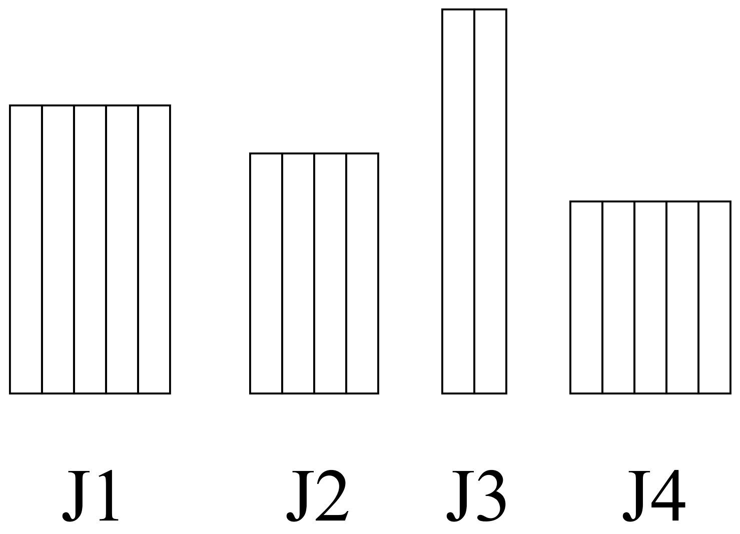
Transformations et ordonnancement

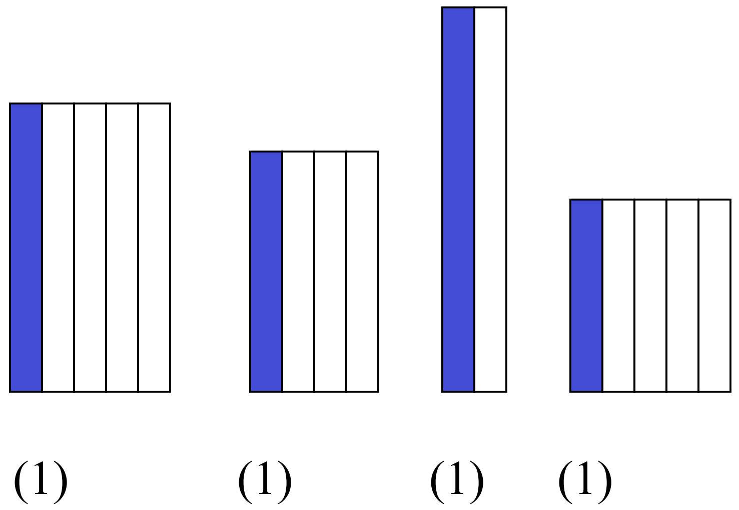


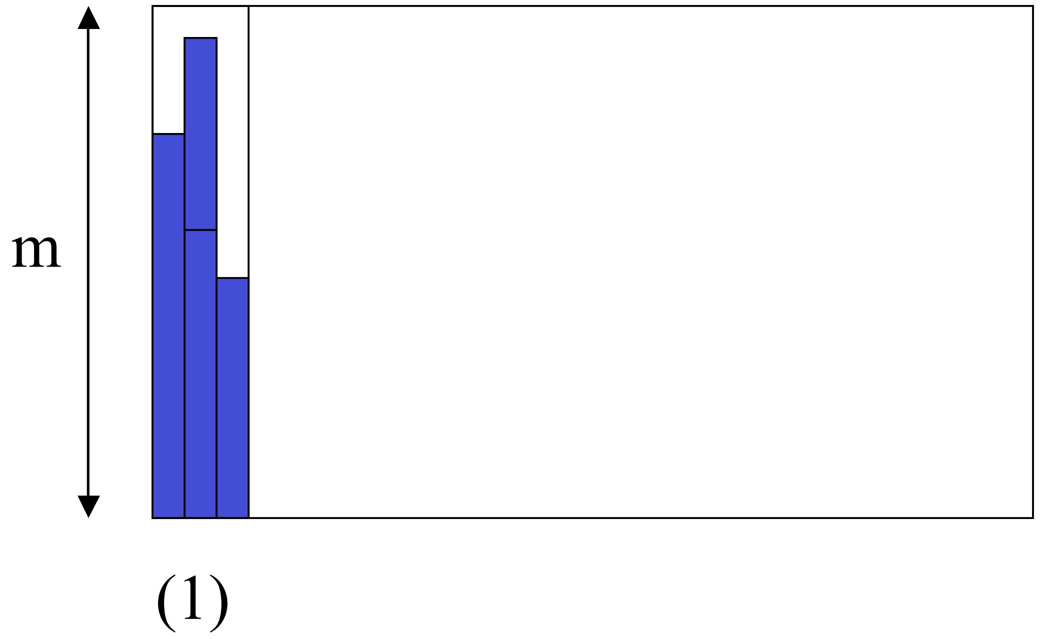
Jobs rigides
(non clairvoyant)

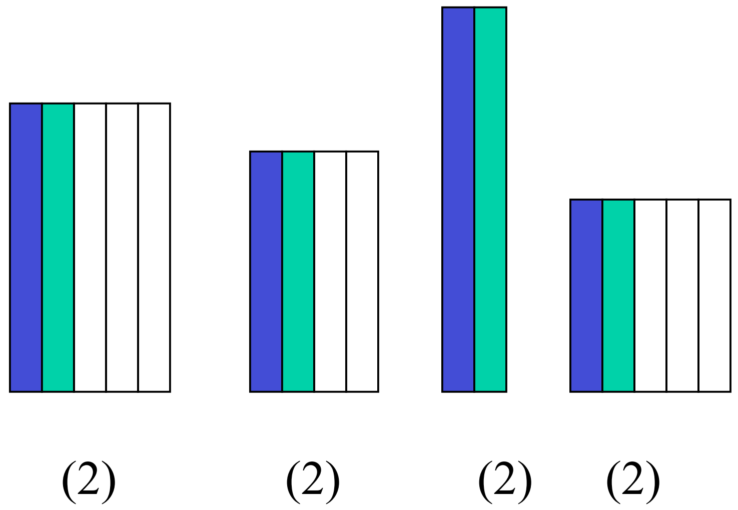
Soit n jobs indépendants à ordonnancer chacun avec une estimation du nombre de processeurs pi.

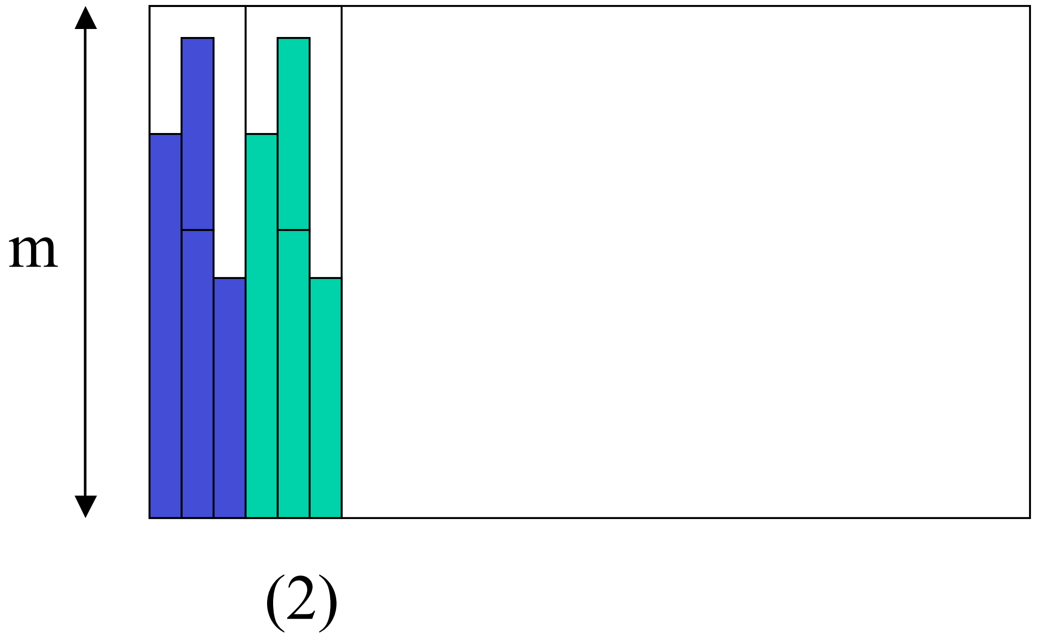
Ordonnancement rigide




(1)
(1)
(1)
(1)


m
(1)


(2)
(2)
(2)
(2)


m
(2)

Performance garantie

4-approximation avec l'heuristique FFD.

Technique : adversaire clairvoyant.

Ordonnancement MT général

Généralisation à des graphes avec structures de précédence : même méthodologie avec un ordonnancement plus complexe (liste avec garantie).

Structures spécifiques (chaines, arbres, SP, …)

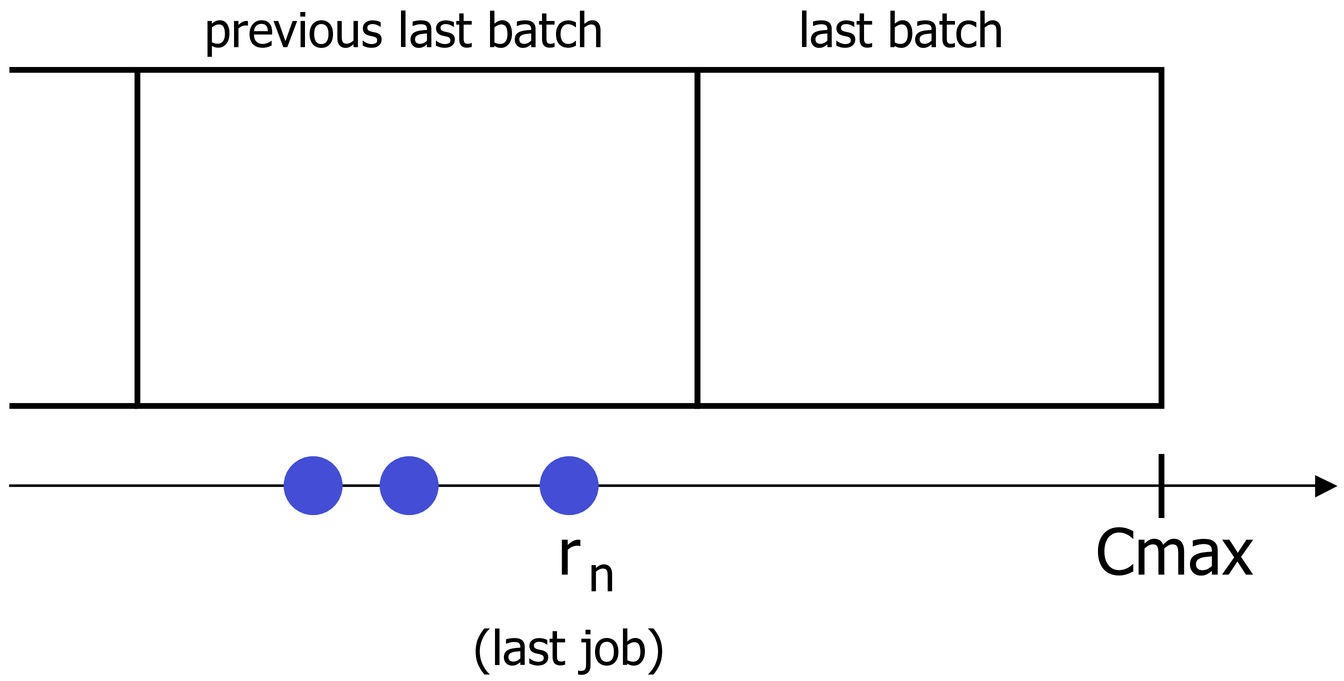
Garanties en (3+sqrt(5))/2

On-line scheduling

Constructing a batch scheduling

Analysis: there exists a nice result which gives a guaranty for an execution in batch function of the guaranty of the scheduling policy inside the batches.

Analysis [Shmoys]



Proposition

$$C_{\max} \leq 2\rho C^*_{\max}$$

Analysis

Tk is the duration of the last batch

$$\rho C^*_{\max} \geq r_n + T_k$$

On another hand, $D_{k-1} \leq r_n$ and $\forall i, T_i \leq \rho C^*_{\max}$

$$C_{\max} = D_{k-1} + T_{k-1} + T_k$$

Thus: $C_{\max} \leq 2\rho C^*_{\max}$

Multi criteria

The Makespan is not always the adequate criterion.

System point of view:

Average completion time (weighted)

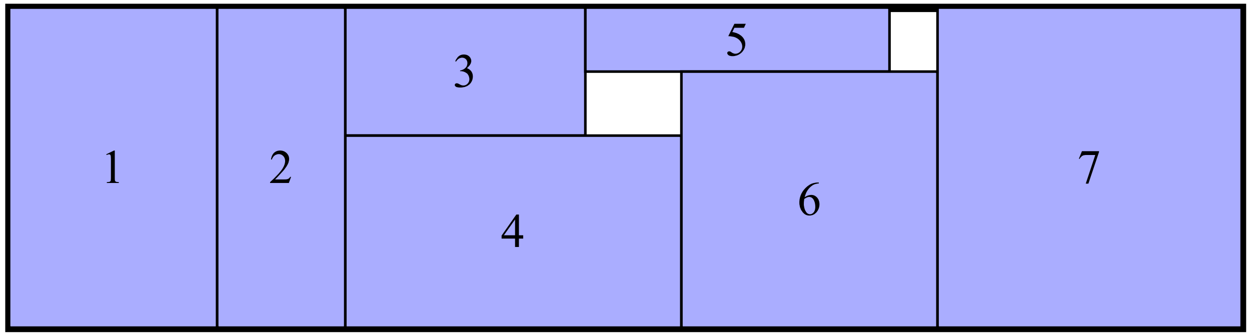
Other criteria: Stretch, Asymptotic throughput

A first solution

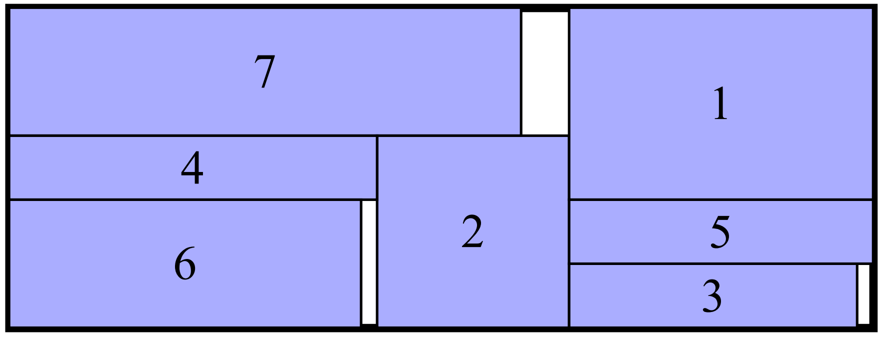
Construct a feasible schedule from two schedules of guaranty r for minsum and r’ for makespan with a guaranty (2r,2r’) [Stein et al.].

Instance: 7 jobs (moldable tasks) to be scheduled on 5 processors.

Schedules s and s’

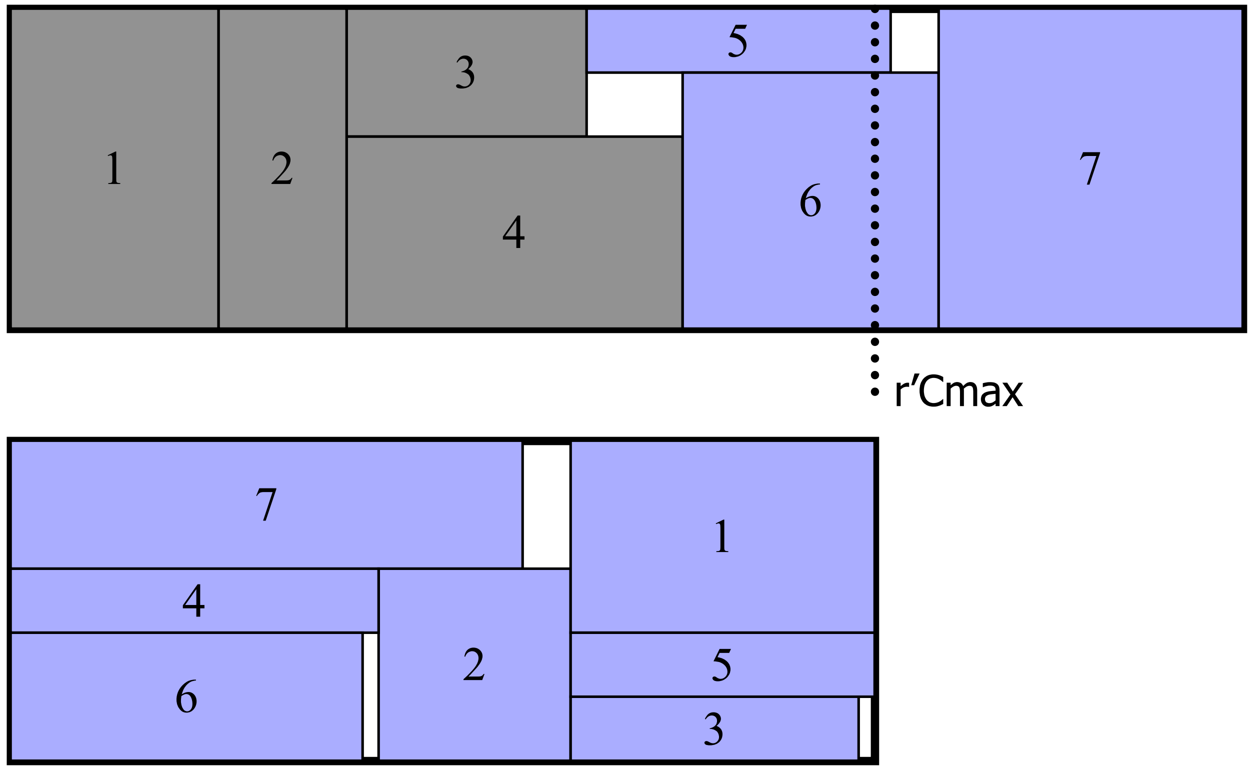


Schedule s
(min sum)

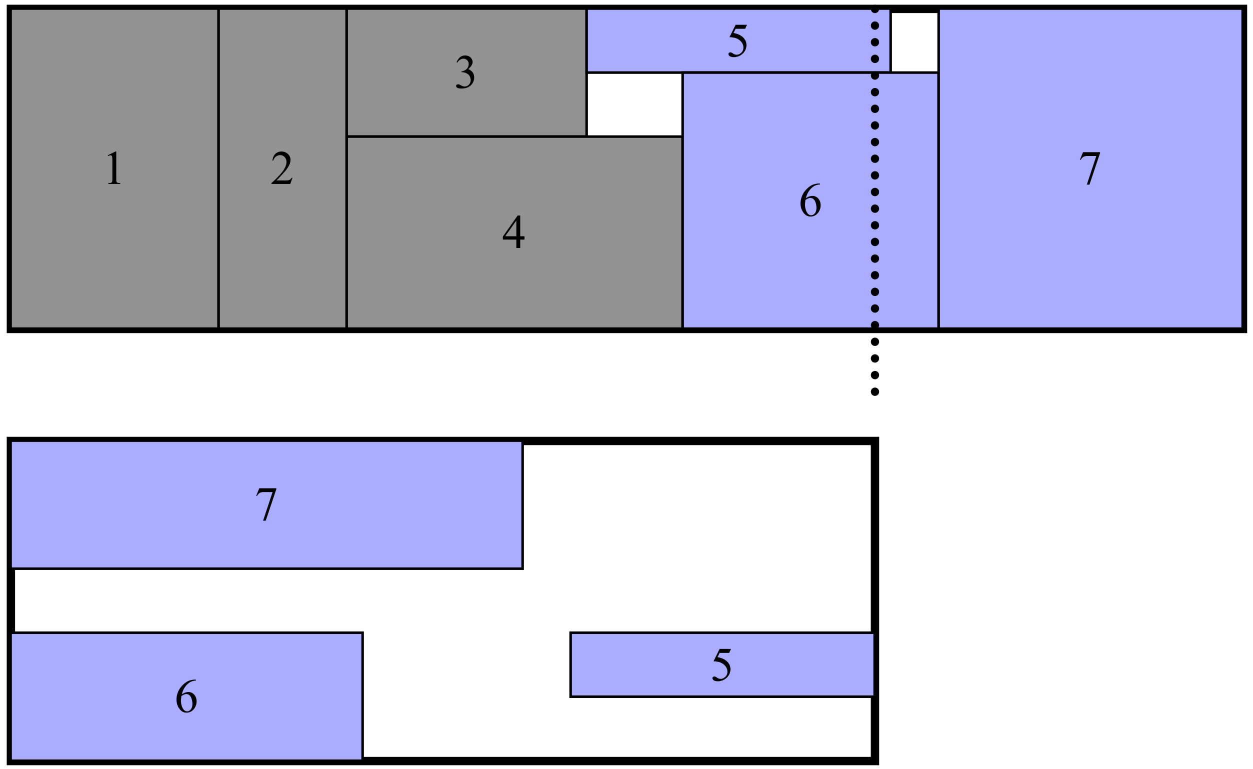


Schedule s’
(makespan)

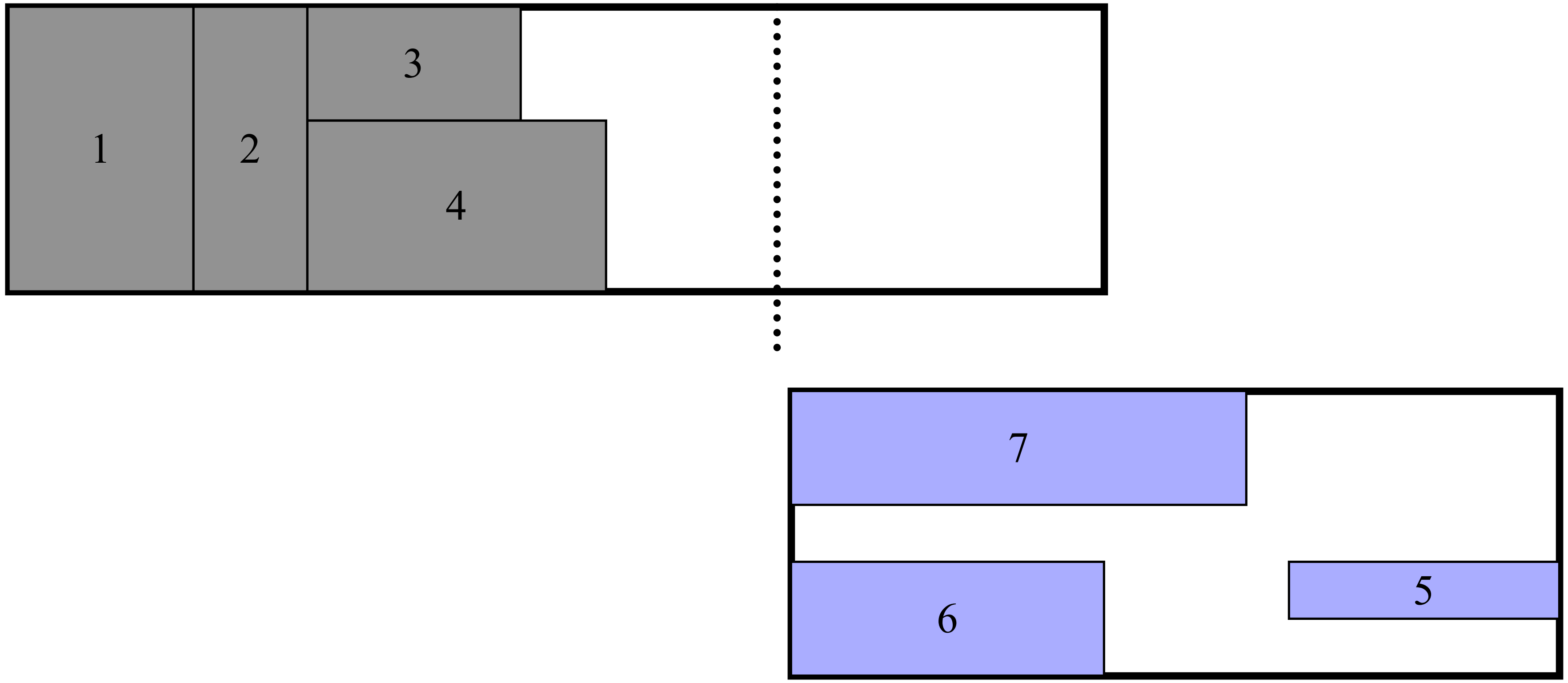
New schedule



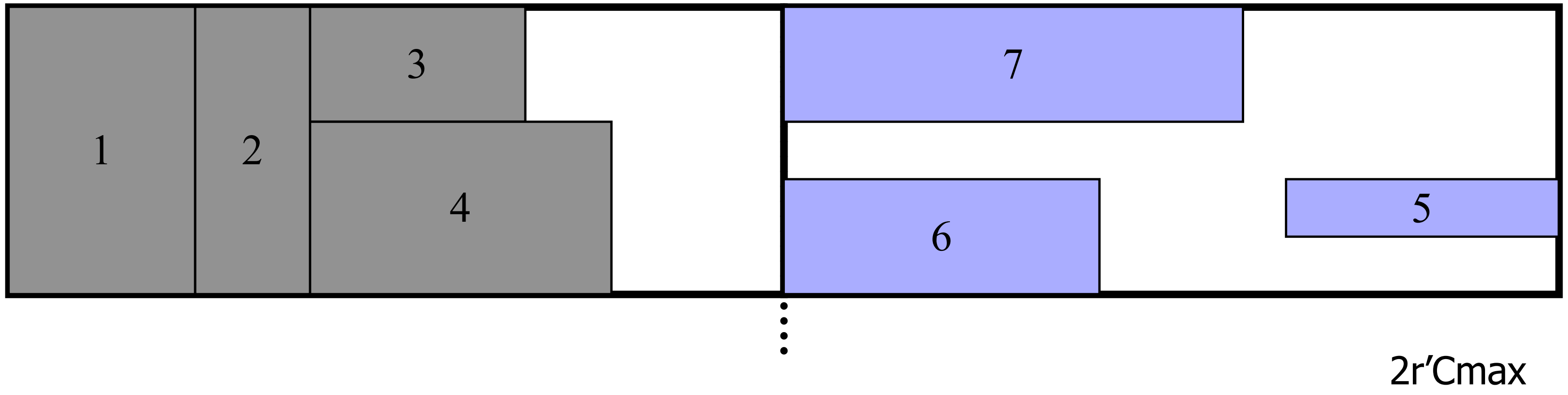
New schedule



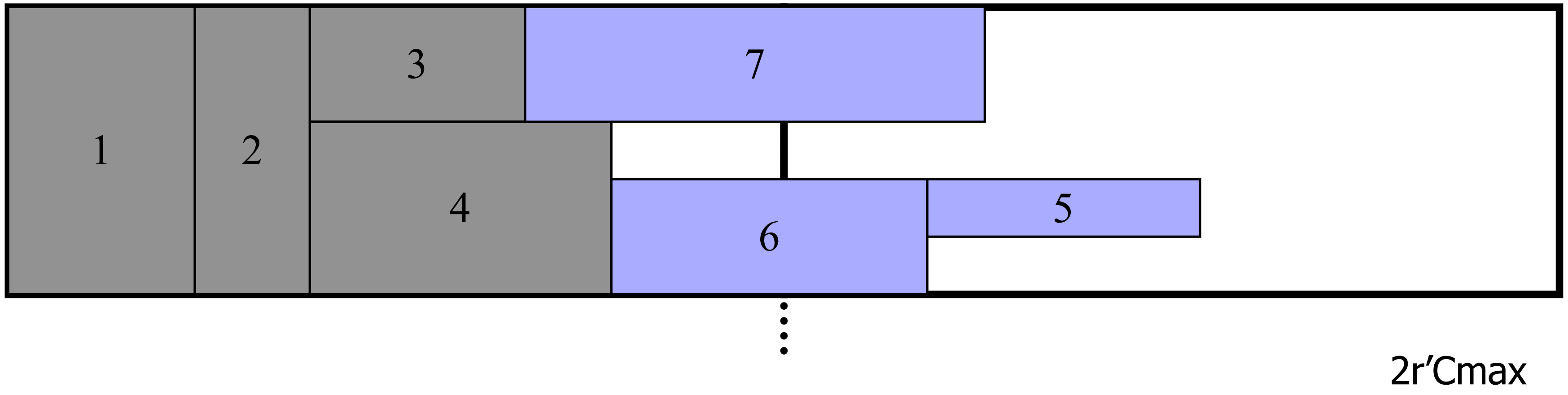
New schedule



New schedule



New schedule



Similar bound for the first criterion

Analysis

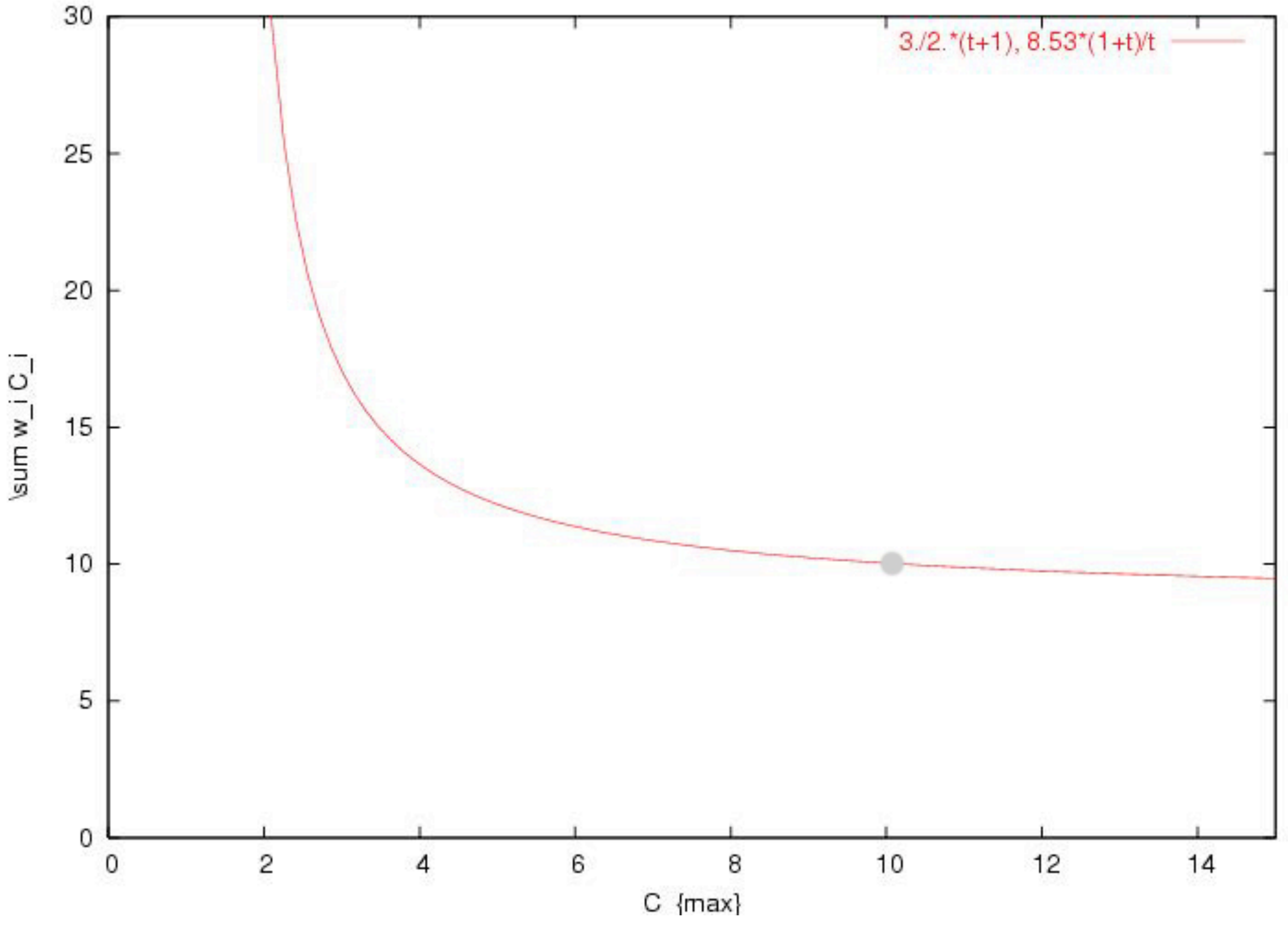
The best known schedules are:
8.53 [Schwiegelsohn] for minsum and 3/2 [Mounie et al.] for makespan leading to (17.06,3).

Improvement

We can improve this result by determining the Pareto curves (of the best compromises):
$(1+\lambda)/ \lambda\, r$ and $(1+ \lambda)r'$

Idea:
take the first part of schedule s up to $\lambda\, r'C_{max}$

Pareto curve



Another way for better schedules

We propose now a new solution for a better result which has no to consider explicitely schedule for minsum (based on a nice dynamic framework).

Conclusion

We have presented and discussed the problem of scheduling in the context of Parallel Processing.

There is an important impact of the computational model on the performances.

Communications are crucial and have to be optimized. Partitioning is more important than internal scheduling.