Efficient Scheduling Algorithms for resource management

Denis Trystram

ID-IMAG

CSC, Toulouse, june 23, 2005

General Context

Recently, there was a rapid and deep evolution of execution supports: super-computers, clusters, grid computing, global computing...

Need of efficient tools for resource management for dealing with these new systems.

This talk will investigate some of the related problems and discuss new solutions

Parallel computing today.

- Different kinds
  - Clusters, collection of clusters, grid, global computing
  - Set of temporary unused resources
  - Autonomous nodes (P2P)
- Our view of grid computing (reasonable trade-off):
  - Set of computing resources under control (no hard authentication problems, no random addition of computers, etc.)

Content

- Goal: to illustrate the scheduling problem in grids
- A first report of experiences in Grenoble
- Intra-clusters scheduling
- On-line
- Multi-criteria
- Inter-cluster scheduling. How to deal with more complex actual problems?

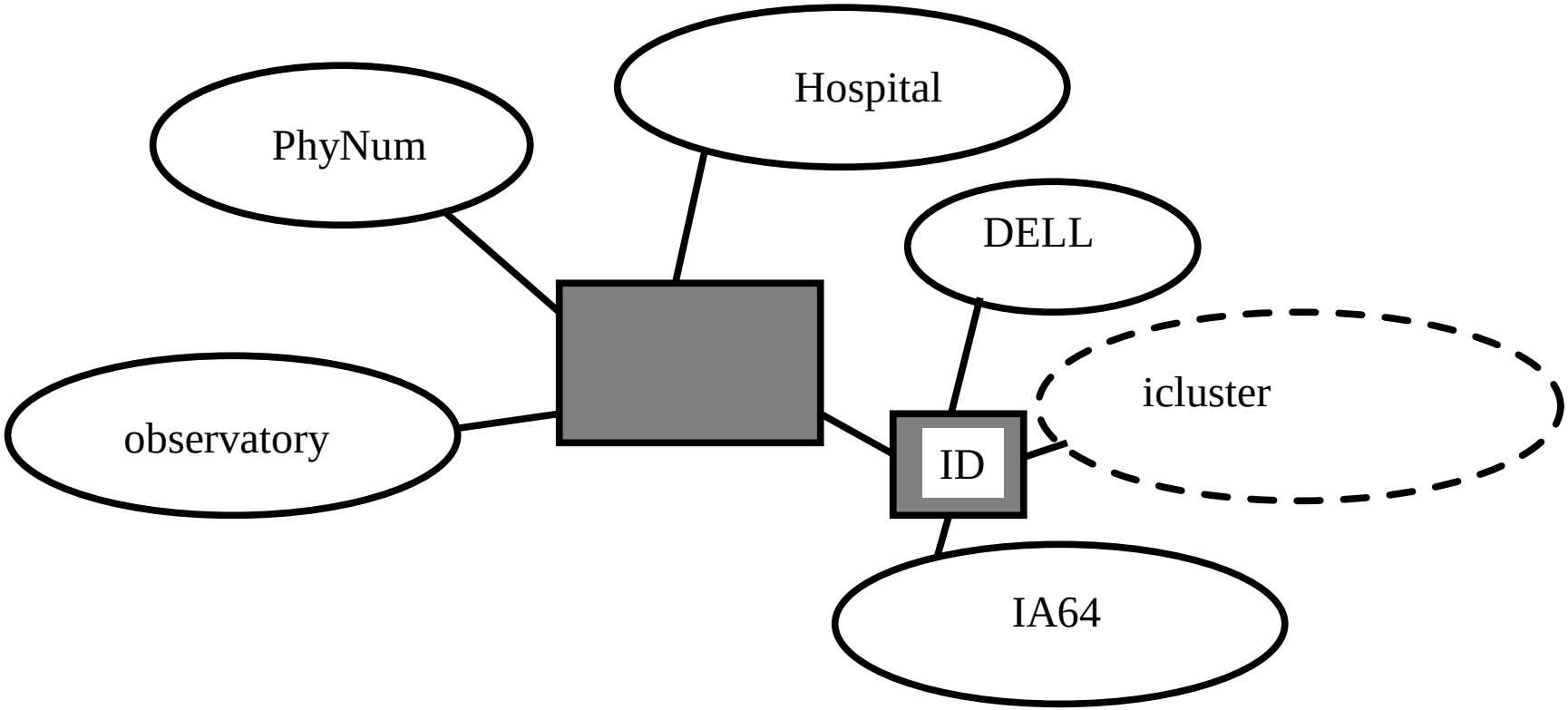
CiGri: a regional grid

It comes from the project CIMENT whose aim was to create new *meso-computing* facilities (alternative to national computing centers) in the late 90ties.

Our view of grids: collection of clusters. Equipment of 5 clusters in different places around Grenoble, local autonomy, but shared experiences and tools (more than 600 connected machines!).

CiGri

Starting in november 2002 (national programme ACI).



Output

Models of applications coming form the real-life.

Synthetic workload generation

Practical tools (OAR + batch scheduling system).

Brief example: Analysis of job characteristics by logs of Icluster

Icluster is a 225 PCs machine from HP with a rather slow hierarchical interconnection module (5x45machines) – fast Ethernet. Processors are PIII 733Mhz it was replaced in 2003 by a 104 dual-processors cluster.

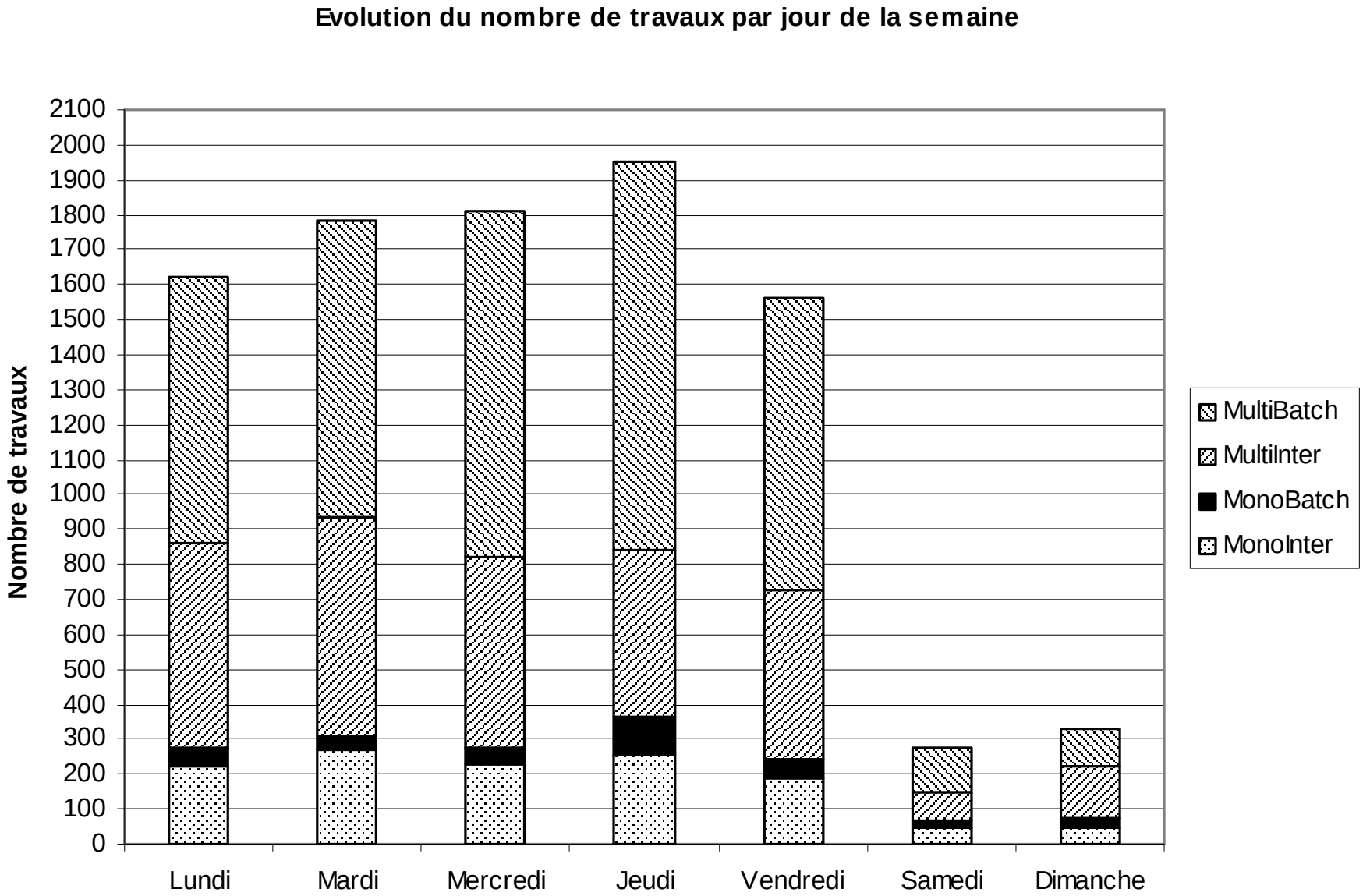
Objectives of the study:

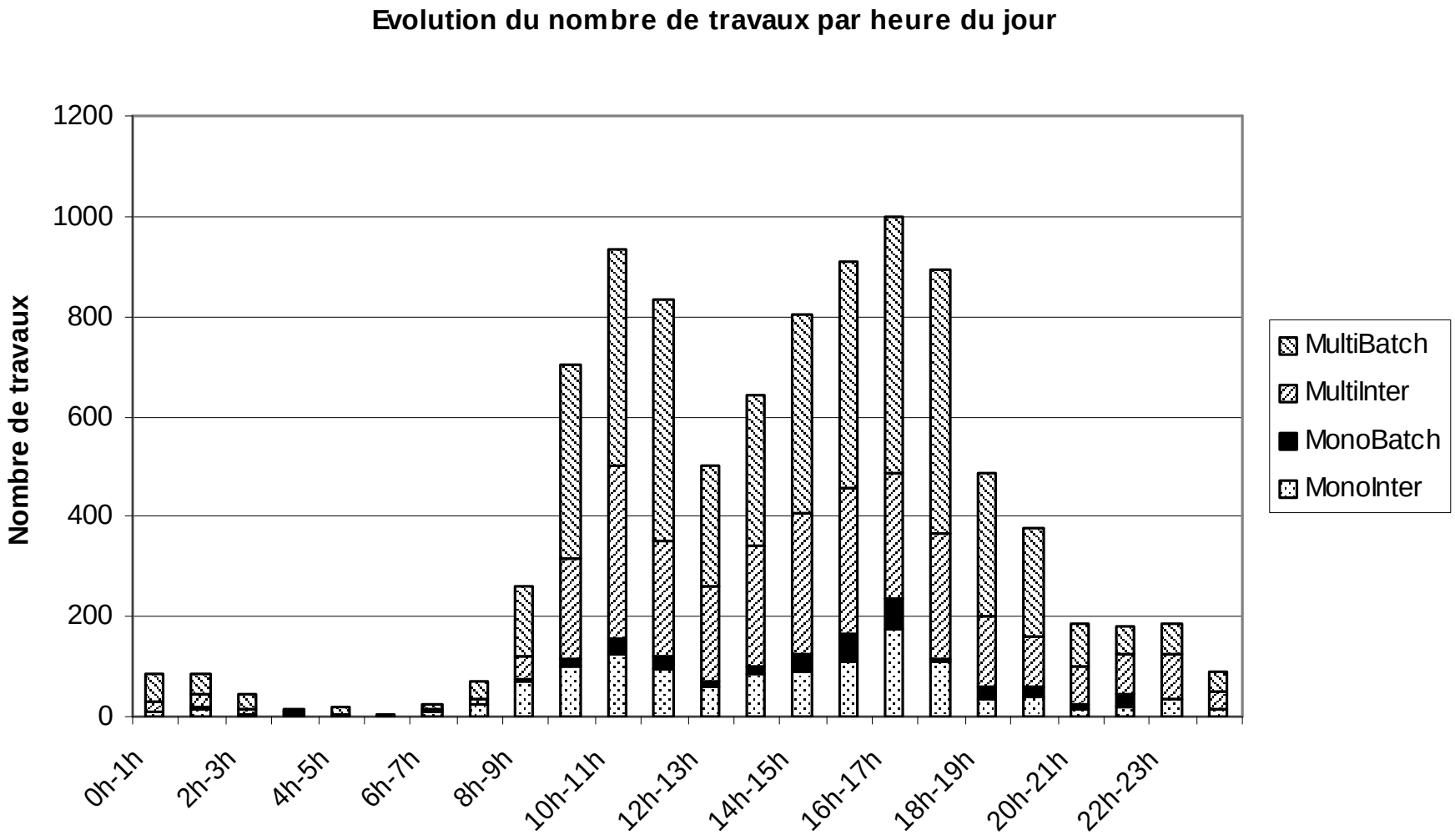
Better understanding of the behaviour

Build a Workload generator

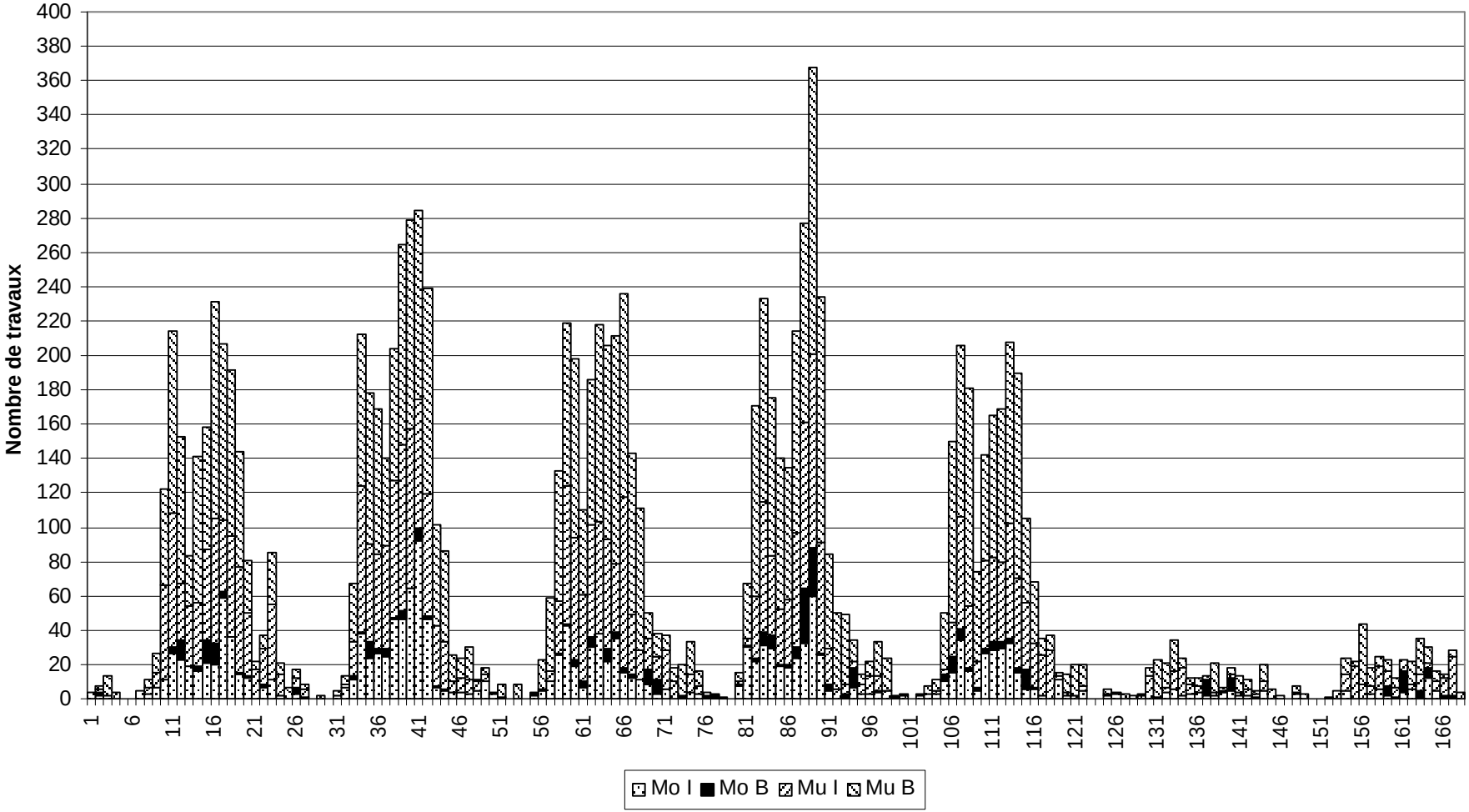
Data (one year)

11496 submissions of all kind of jobs (interactive ones or not, mono and multi-processors). 85 users.
About 10% of these jobs are coming from the system team and were removed from the analysis.


Evolution du nombre de travaux par jour de la semaine
Nombre de travaux
2100
2000
1900
1800
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
0
Lundi
Mardi
Mercredi
Jeudi
Vendredi
Samedi
Dimanche
MultiBatch
MultiInter
MonoBatch
MonoInter


Evolution du nombre de travaux par heure du jour
Nombre de travaux
1200
1000
800
600
400
200
0
0h-1h
2h-3h
4h-5h
6h-7h
8h-9h
10h-11h
12h-13h
14h-15h
16h-17h
18h-19h
20h-21h
22h-23h
MultiBatch
MultiInter
MonoBatch
MonoInter

Evolution du cumul de travaux par heure de la semaine



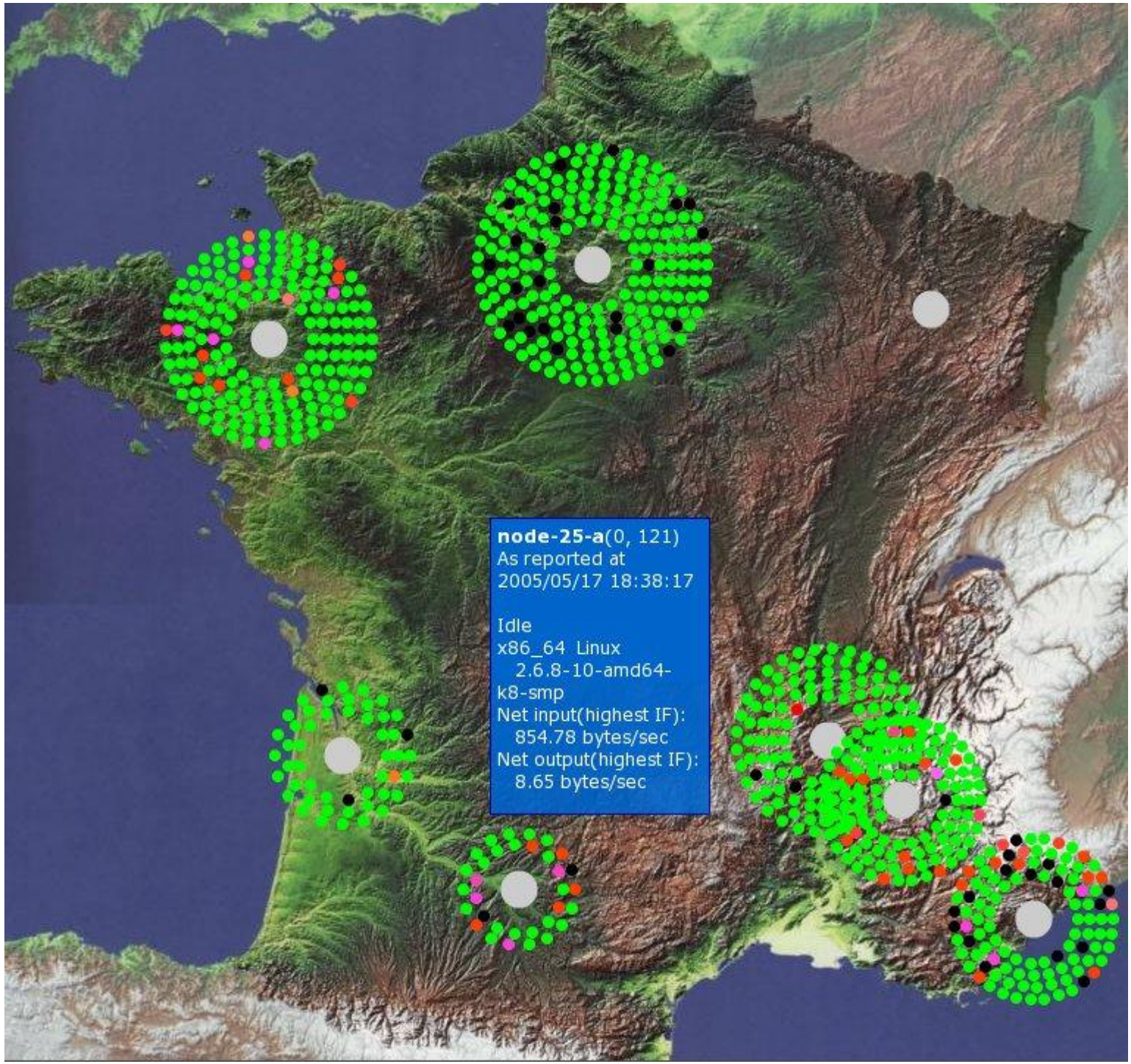
Synthetic Generation of workloads

Based on these logs, we derived the probabilistic laws of arrivals.

Percentage of jobs using power of 2 (23) or multiple of ten (17) processors.

Workload characterization using Dowley's model.

A new national french initiative: GRID5000

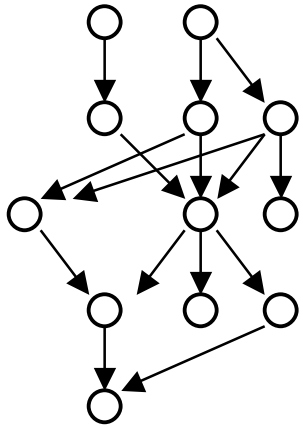

node-25-a(0, 121)
As reported at
2005/05/17 18:38:17
Idle
x86_64 Linux
2.6.8-10-amd64-
k8-smp
Net input(highest IF):
854.78 bytes/sec
Net output(highest IF):
8.65 bytes/sec

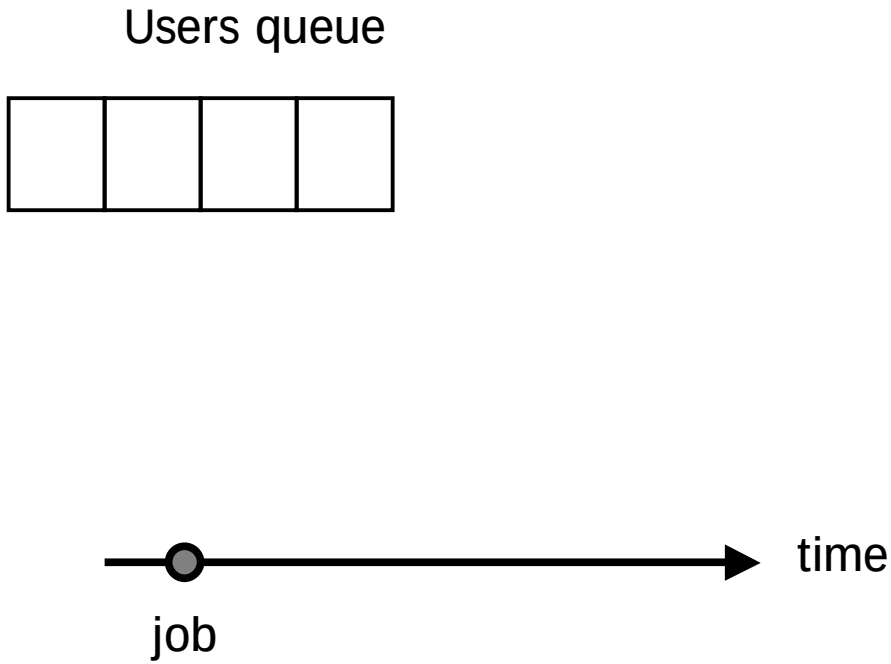
Target Applications

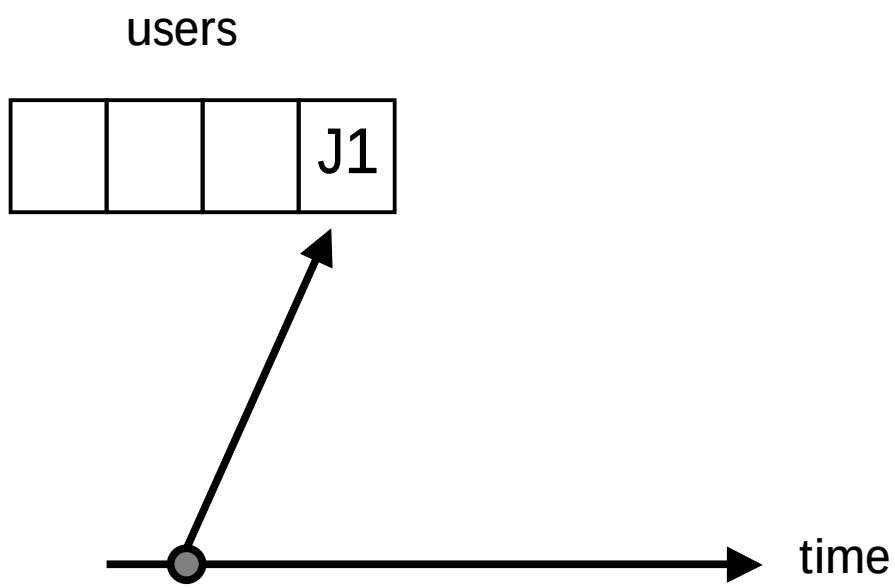
New execution supports created new applications (data-mining, bio-computing, coupling of codes, interactive, virtual reality, ...).

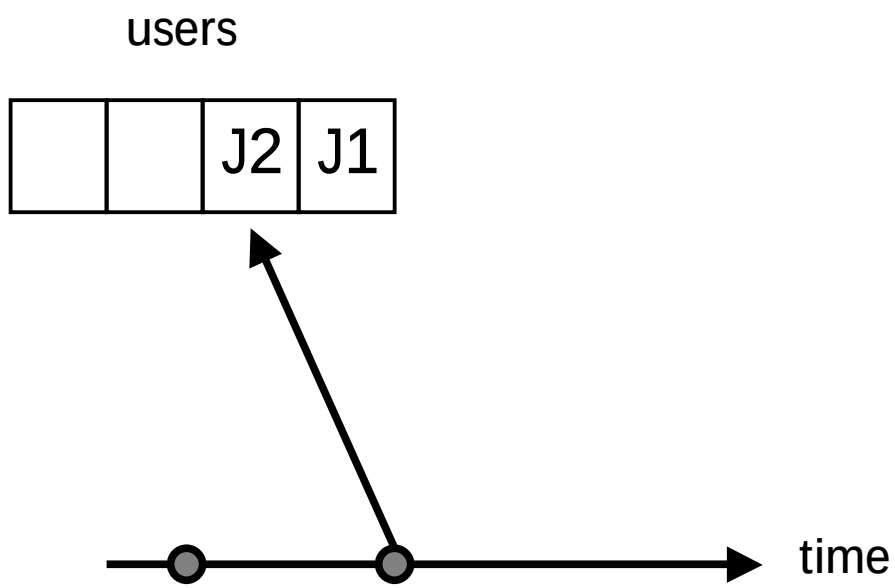
Interactive computations (human in the loop), adaptive algorithms, etc..

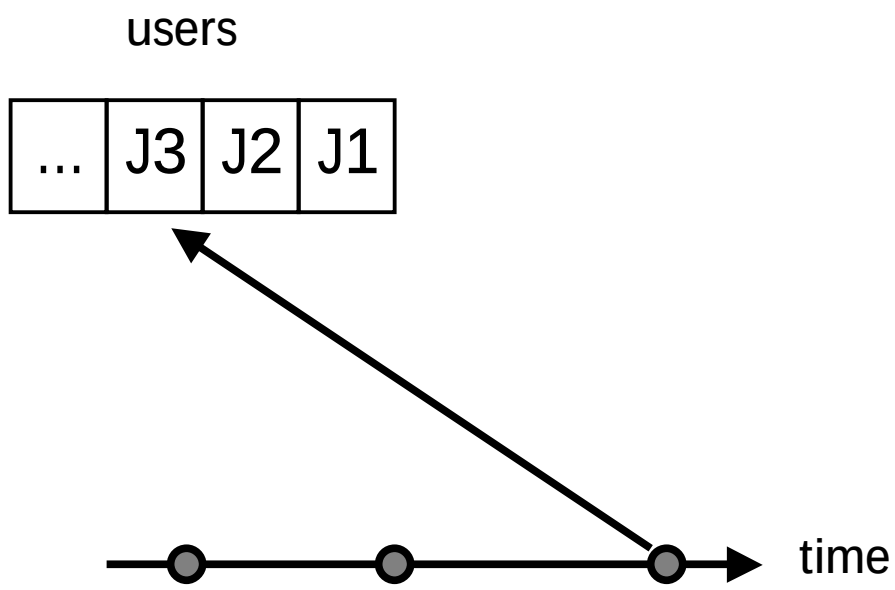
J1

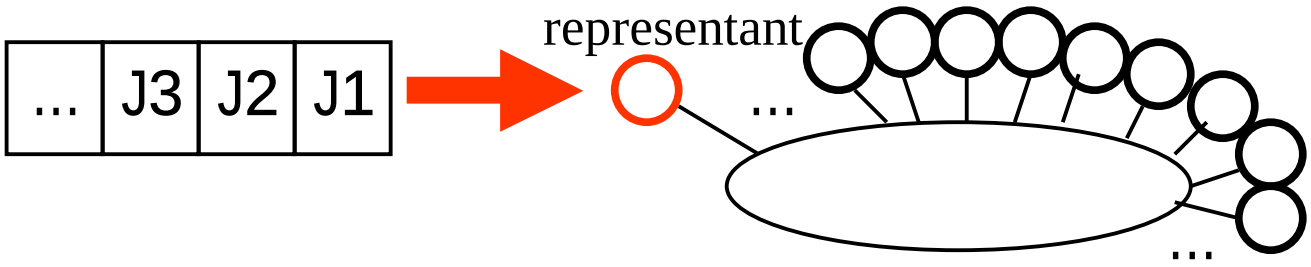


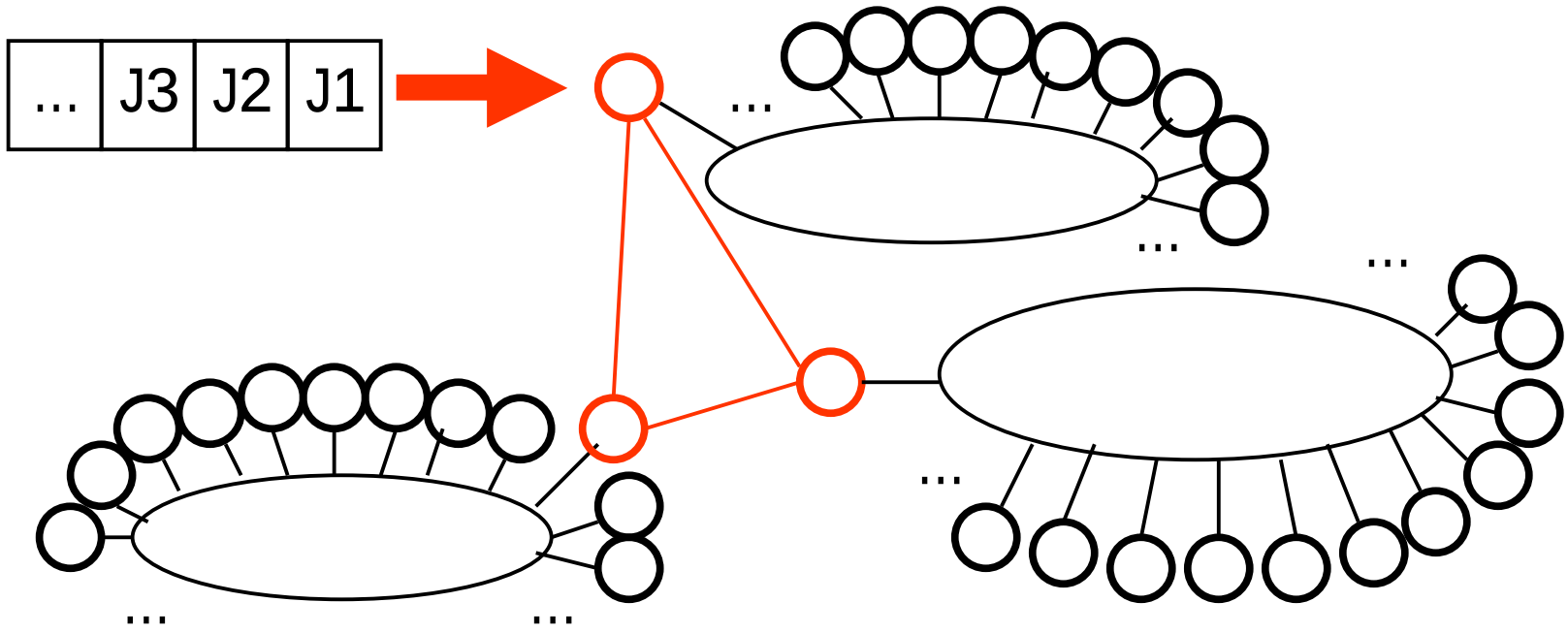

Users queue
time
job


users
J1
time

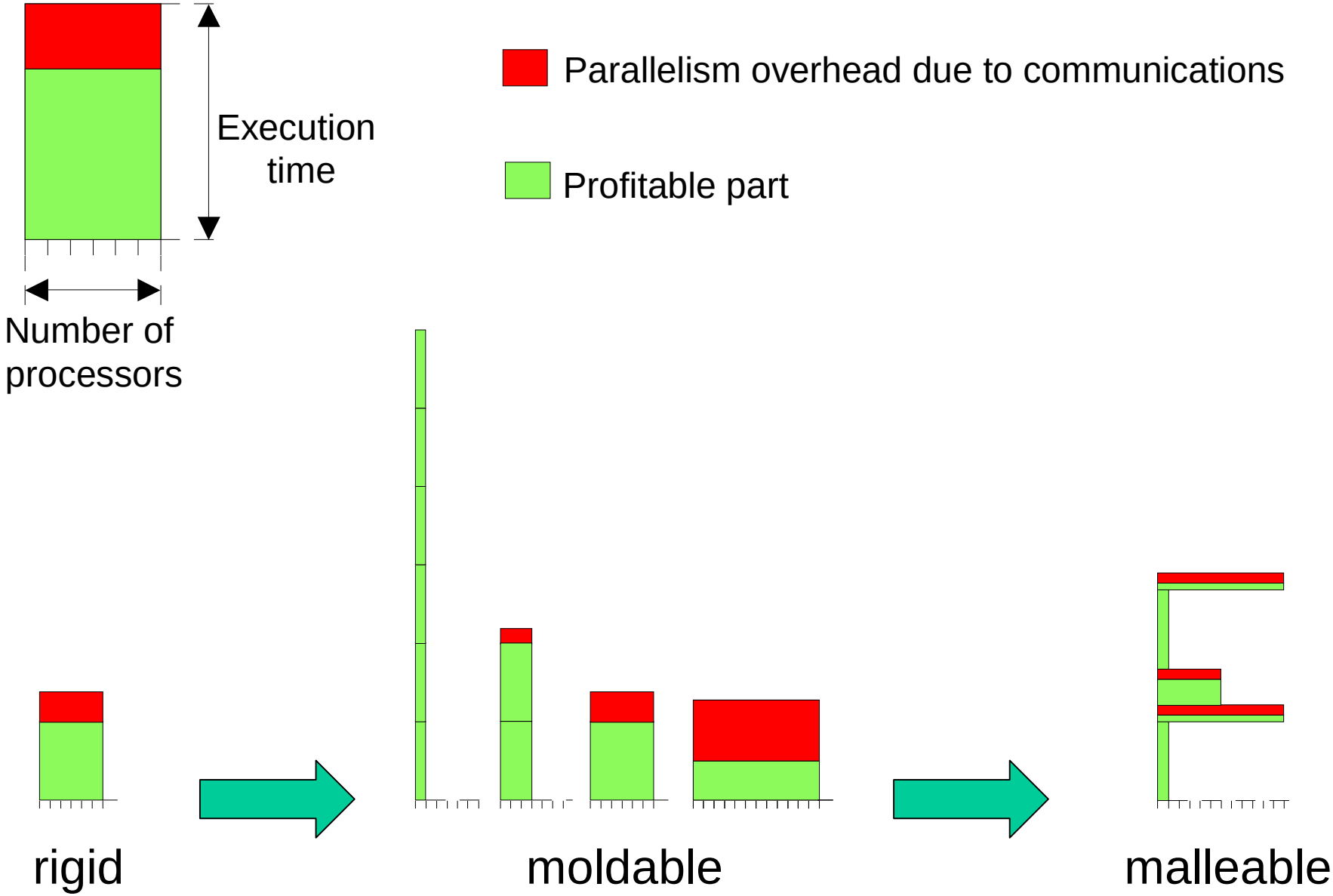

users
J2
J1
time


users
...
J3
J2
J1
time


...
J3
J2
J1
representant
...
...


...
J3
J2
J1

Scheduling: kind of jobs



Basics in scheduling

Central scheduling problem

- Parameters: number and types of processors, structure of the application, criterion to optimize.

Central Scheduling Problem

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

Thus, we are looking for good heuristics.

Analysis using Competitive ratio r:

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

The schedule σ is said r -competitive iff $r(s) \leq r$

Formal Definition

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

(with communication delays)

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

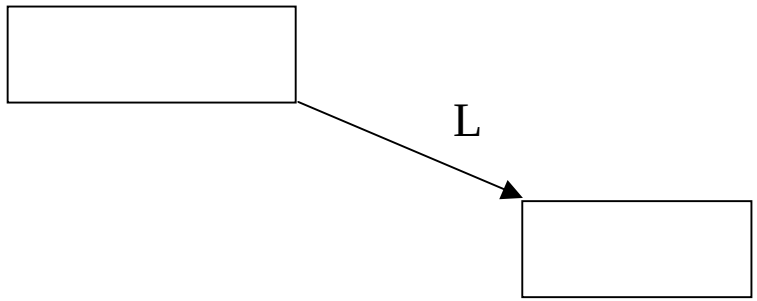
•respect of precedence constraints

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

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

Trivial remark: allocation is important even while scheduling on identical processors…

If L is large, the problem is very hard (no approximation)



Taking into account heterogeneity

More complicated models: LogP

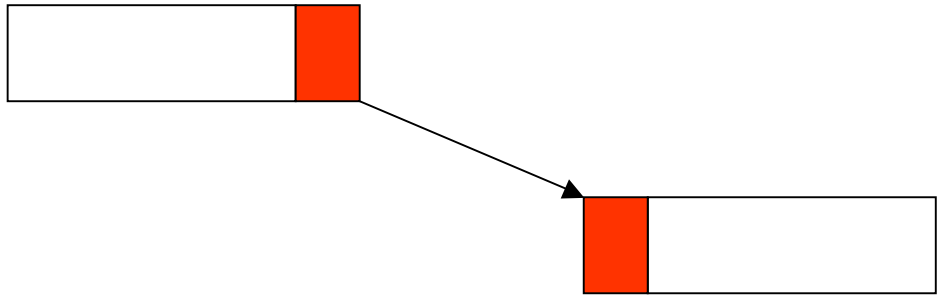
Tentative of designing new computational models closer to the actual parallel systems [Culler et al.]:

4 parameters.

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

Alternative models: LogP

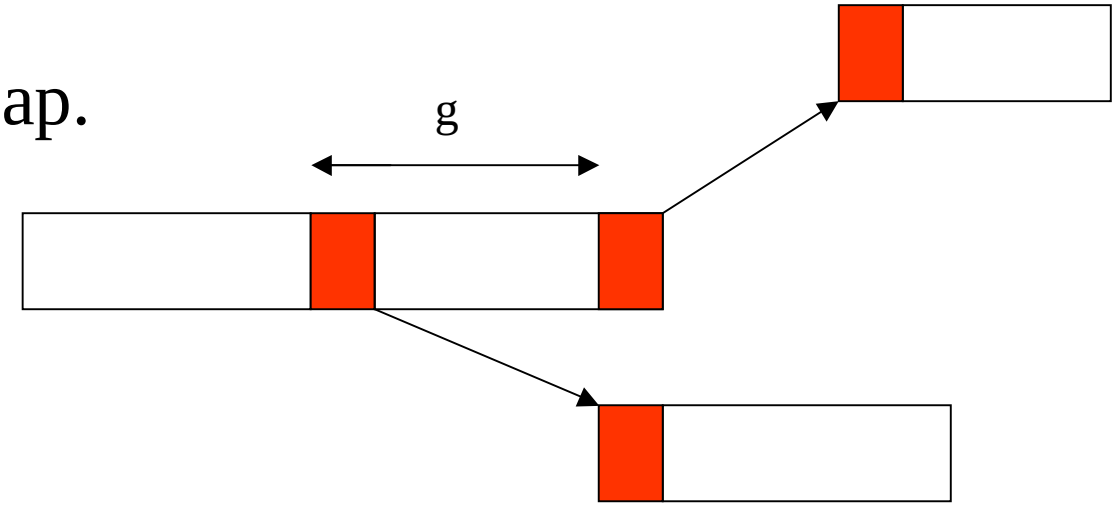
No overlap.



O + L + O

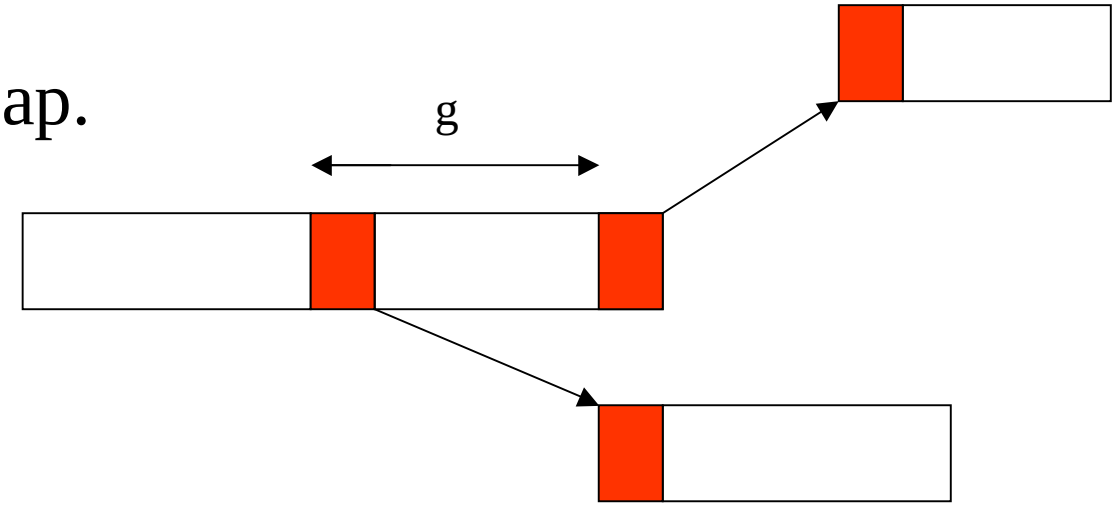
Alternative models: LogP

No overlap.



Alternative models: LogP

No overlap.



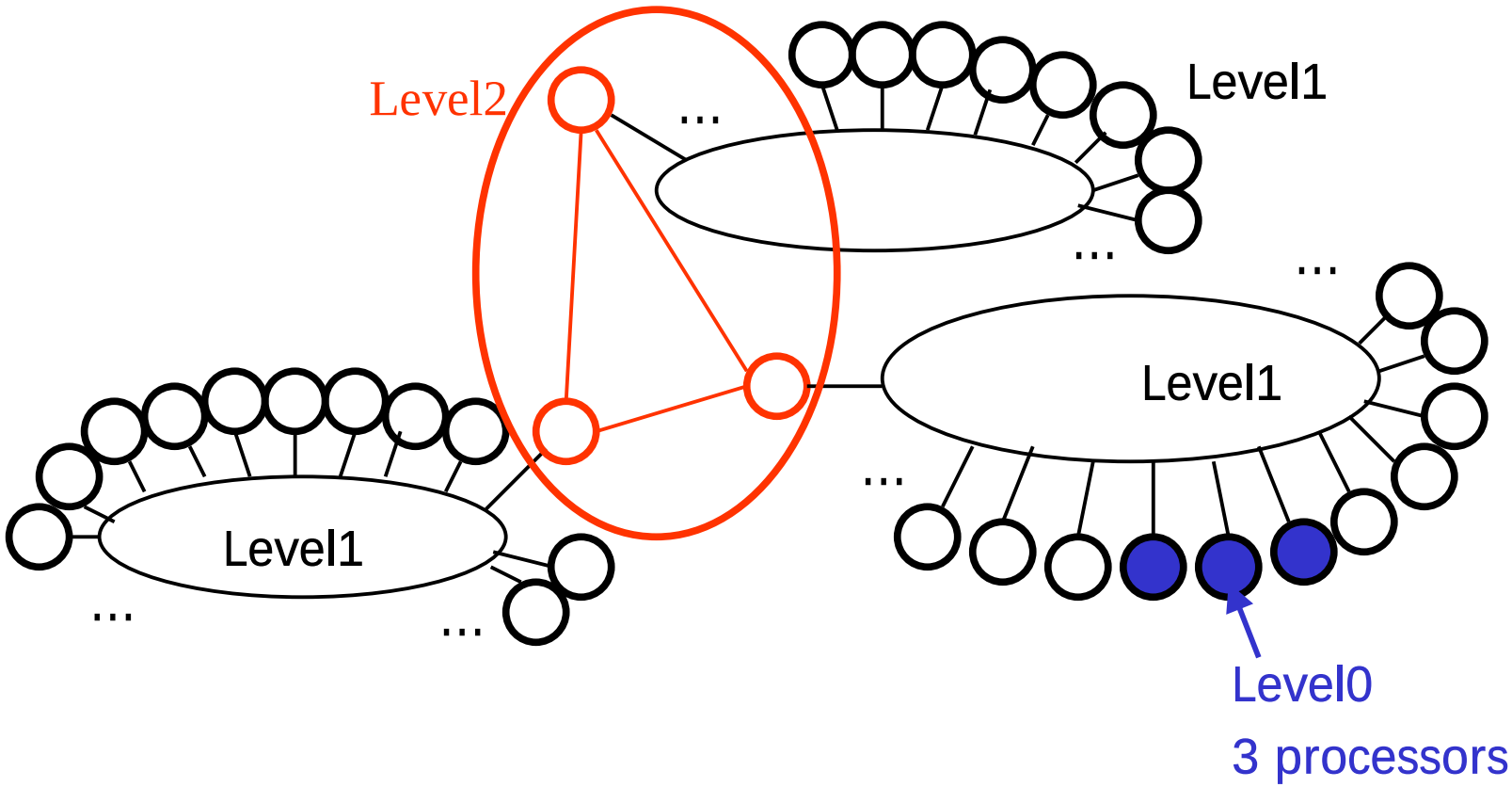
The delay model is a LogP-system where o=g=0

Need to simplify the model

The game becomes too complicated

No way for finding good approximation algorithms (more and more negative results are available...):

Parallel tasks focus on key optimization parameters... No need of sophisticated analysis at the finest grain!


Level2
Level1
Level1
Level1
Level0
3 processors

Intra-Cluster Scheduling

Context :

On-line scheduling

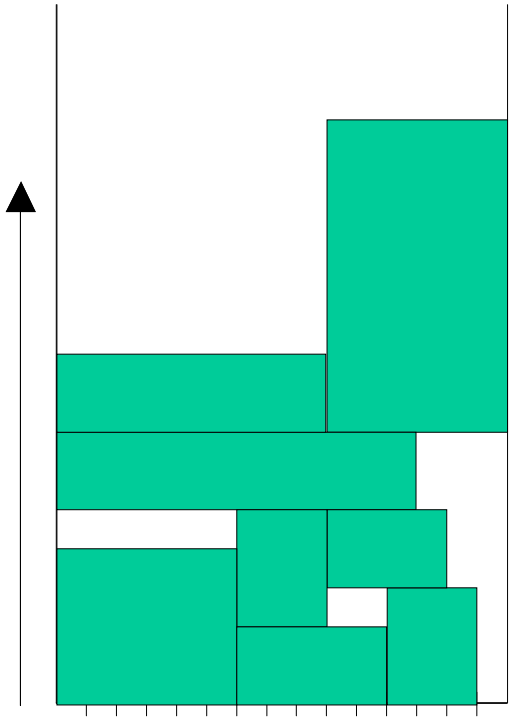
independent jobs (applications represented as moldable – non rigid - tasks) are submitted to the scheduler at any time on a queue.

(strip) Packing problems

The schedule is divided into two successive steps:

1. Allocation problem and
2. Scheduling with preallocation (NP-hard in general [Rayward-Smith 95]).

Scheduling: on-line vs off-line



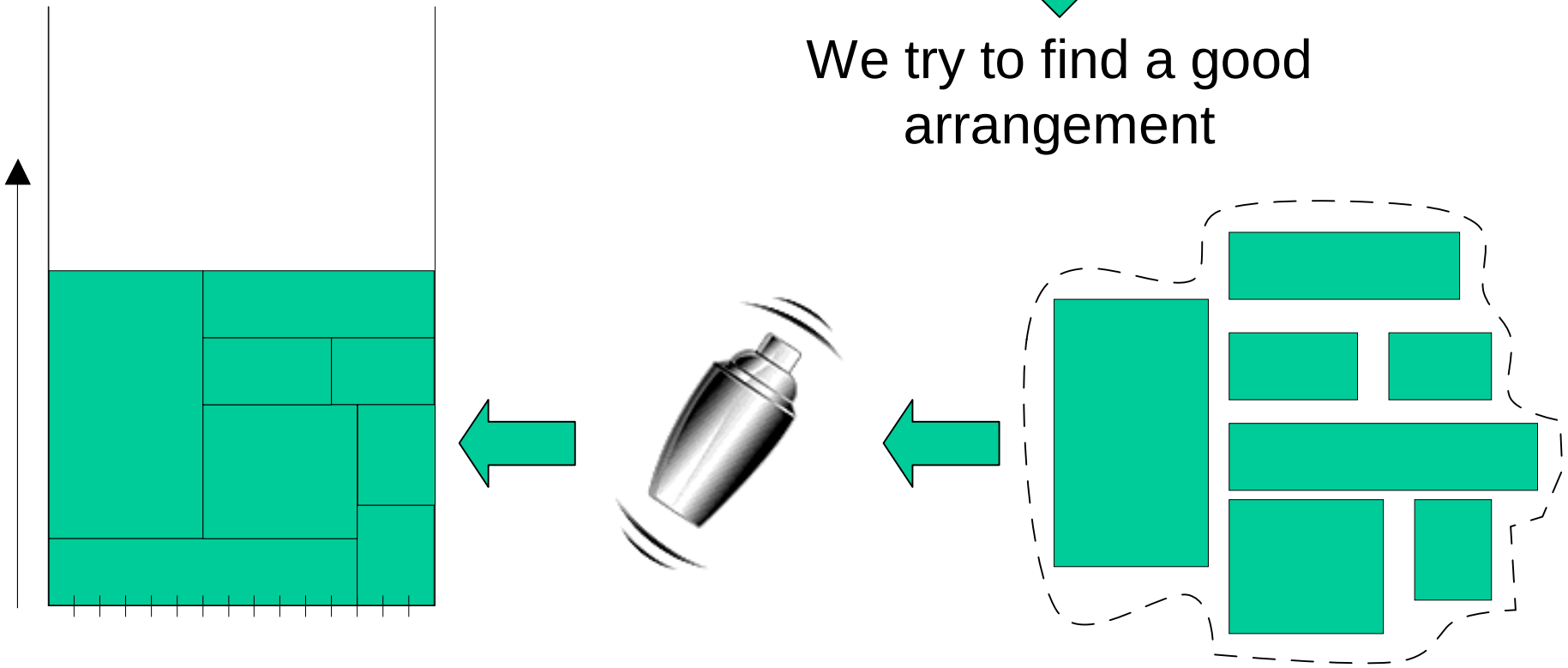
On-line: no knowledge about the future

We take the scheduling decision while other jobs arrive

Scheduling: on-line vs off-line

Off-line: we have a finite set of works

We try to find a good arrangement



Off-line scheduler

Problem:

Schedule a set of independent moldable jobs (clairvoyant).

Penalty functions have somehow to be estimated (using complexity analysis or any prediction-measurement method like the one obtained by the log analysis).

Classical approach using combinatorial optimization

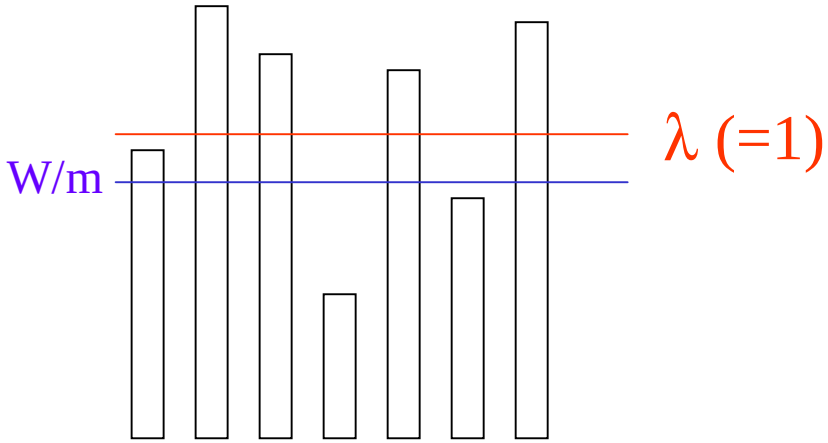
- Design fast algorithms with performance guaranty
- On-line algorithms
- One or several objective functions

k-dual Approximation [Shmoys, Hochbaum]

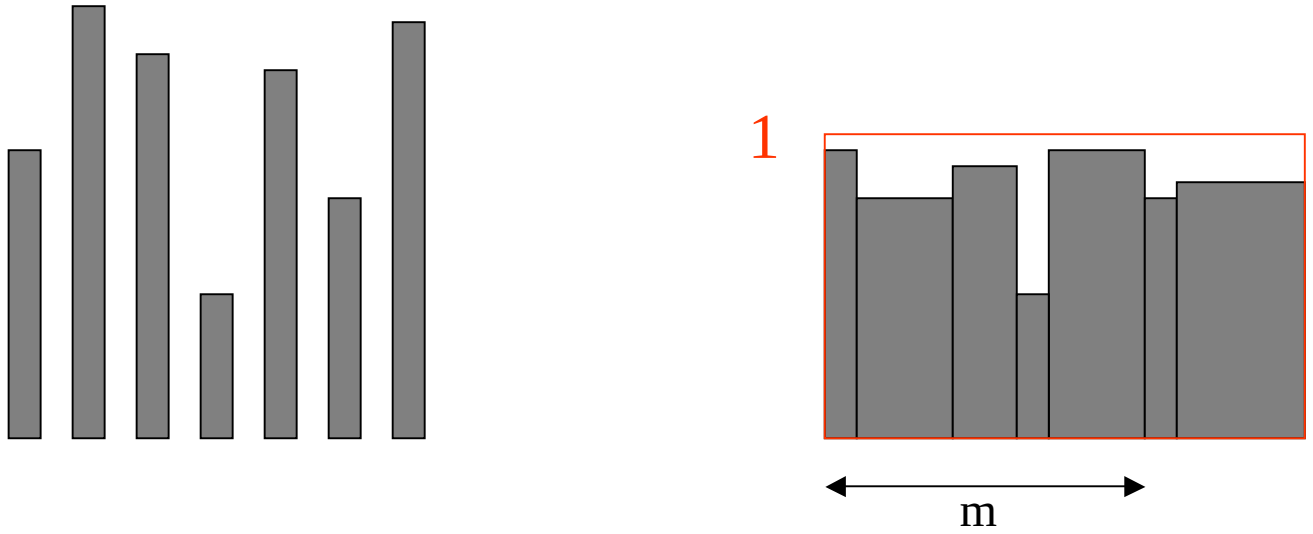
- Let us guess a value of the objective Cmax: λ.
- Apply an algorithm, if the obtained a guaranty worse than λk, then, refine the value of λ (by dichotomic search).

Dual approximation

Estimated a target using a lower bound



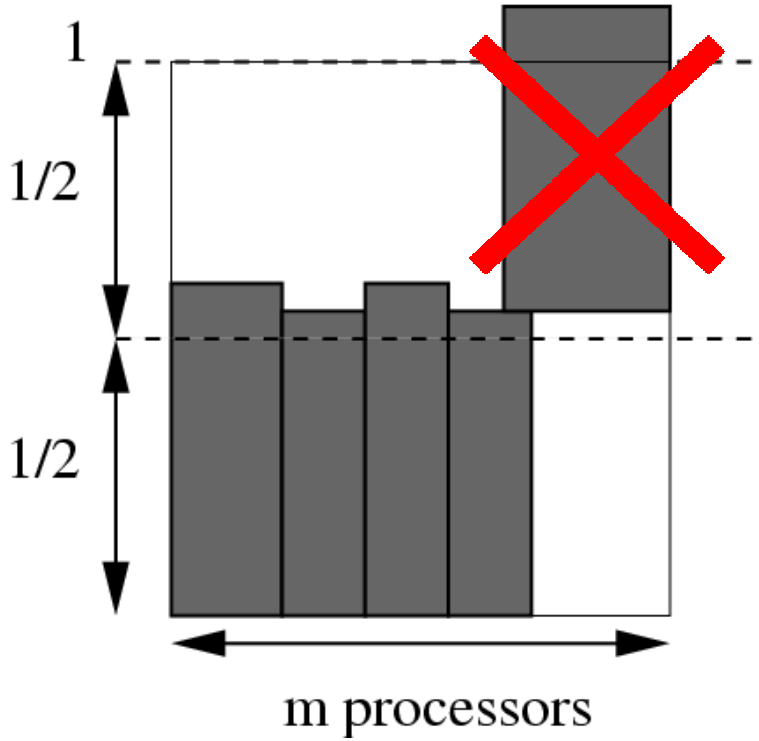
Canonical allotment



maximal number of processors for executing a task in time lower than 1 unit (normalized).

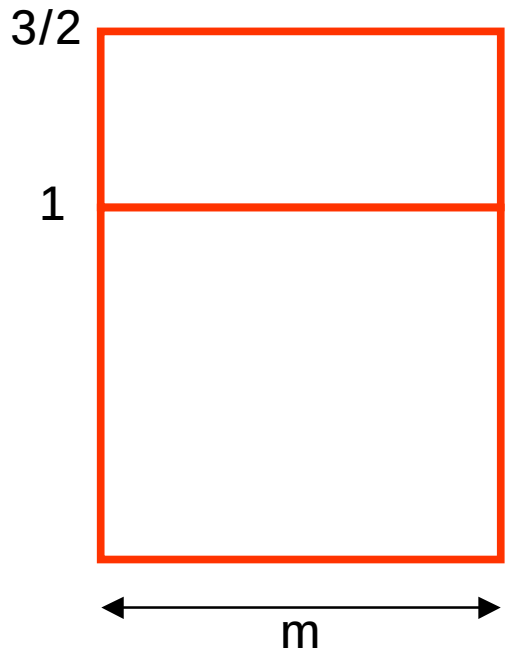
Jobs are assumed to be monotonic.

HINT: analyze the optimal structure

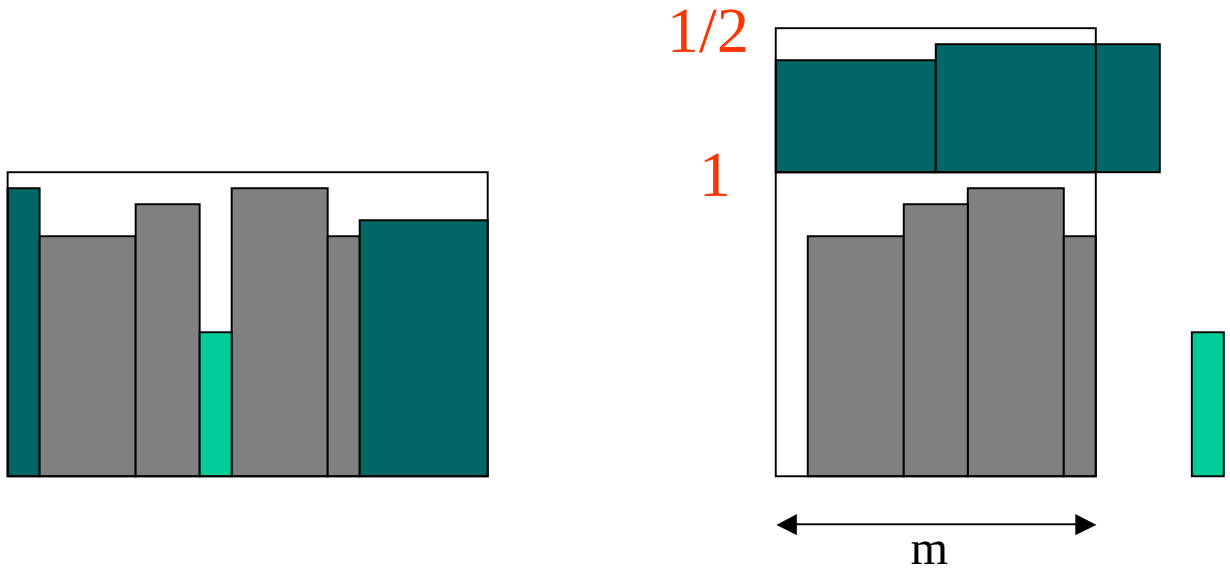


Long Jobs (whose execution times are greater than 1/2) may not use more than m processors

Thus, we are looking for a schedule in two shelves



2 shelves partitioning



Knapsack problem: minimizing the global surface under the constraint of using less than m processors in the first shelf.

Dynamic programming

For i = 1..n // # of tasks

for j = 1..m // #proc.

Wi,j = min(

- Wi,j-minalloc(i,1) + work(i,minalloc(i,1))
- Wi,j + work(i,minalloc(i,1))

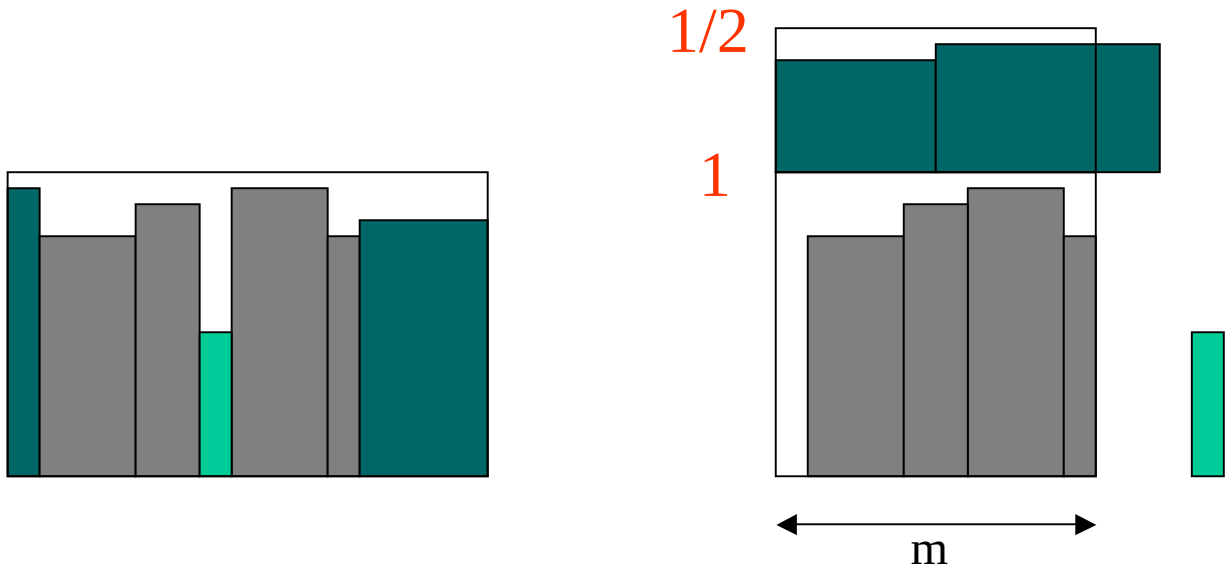
)

work Wn,m

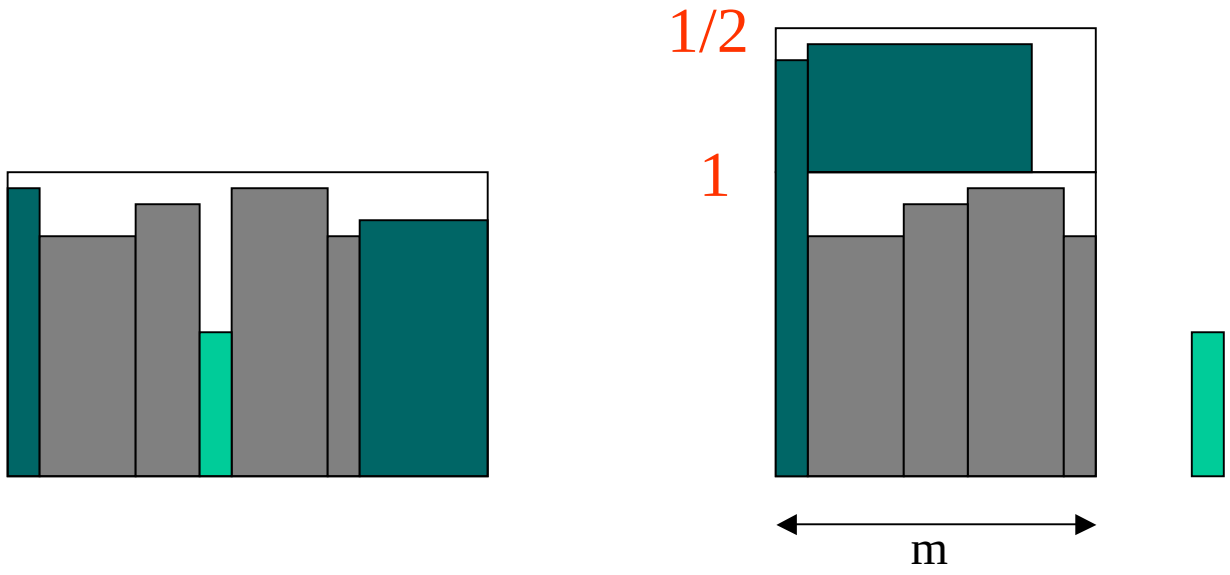
<= work of an optimal solution

but the half-sized shelf may be overloaded

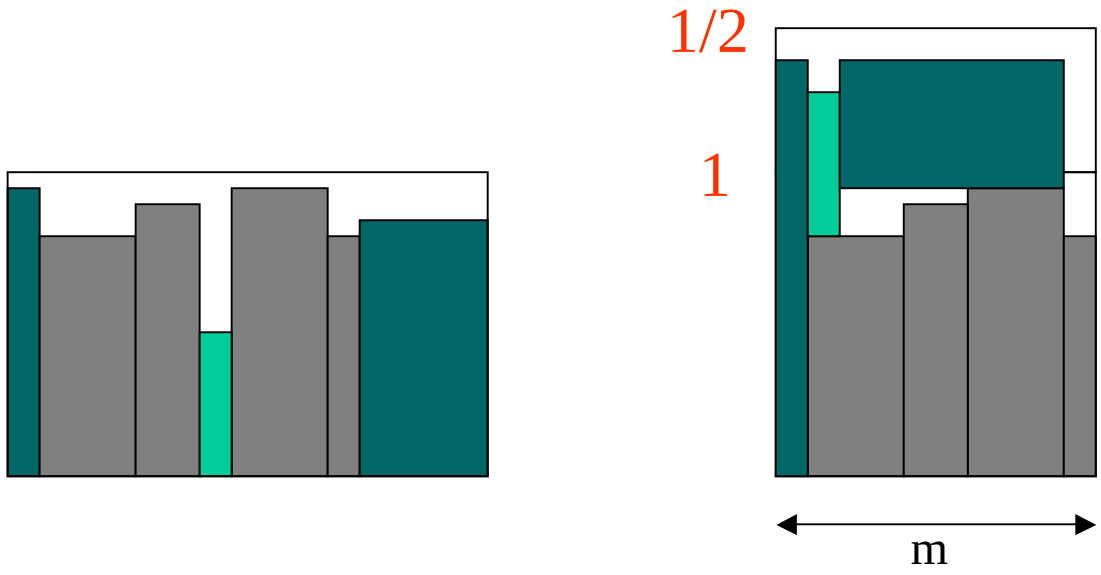
2 shelves partitioning



Drop down



Insertion of small tasks



Analysis

- These transformations donot increase the work
- If the 2nd shelf is used more than m, it is always possible to do one of the transformations (using a global surface argument)
- It is always possible to insert the « small » sequential tasks (again by a surface argument)

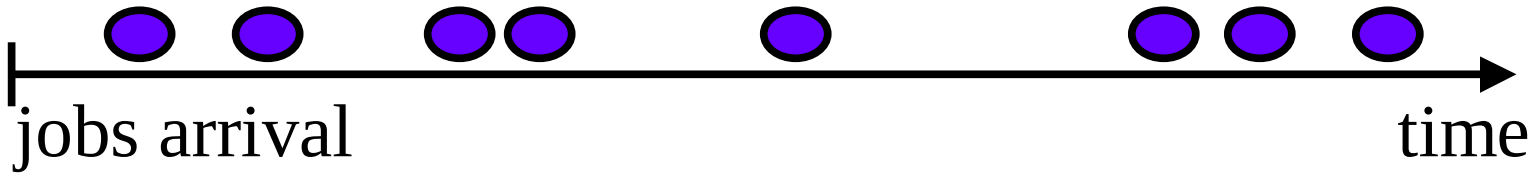
Guaranty

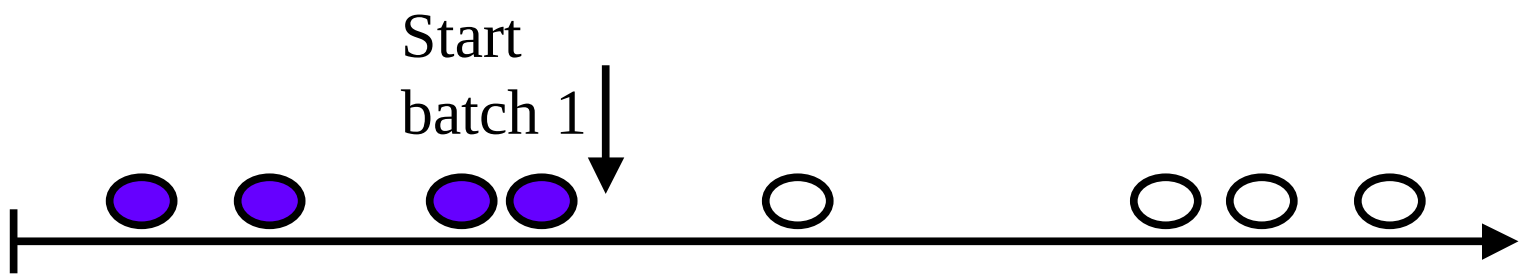
- The 2-shelves algorithm has a performance guaranty of $3/2+\varepsilon$
- We will use it as a basis for an on-line version (batch scheduling).

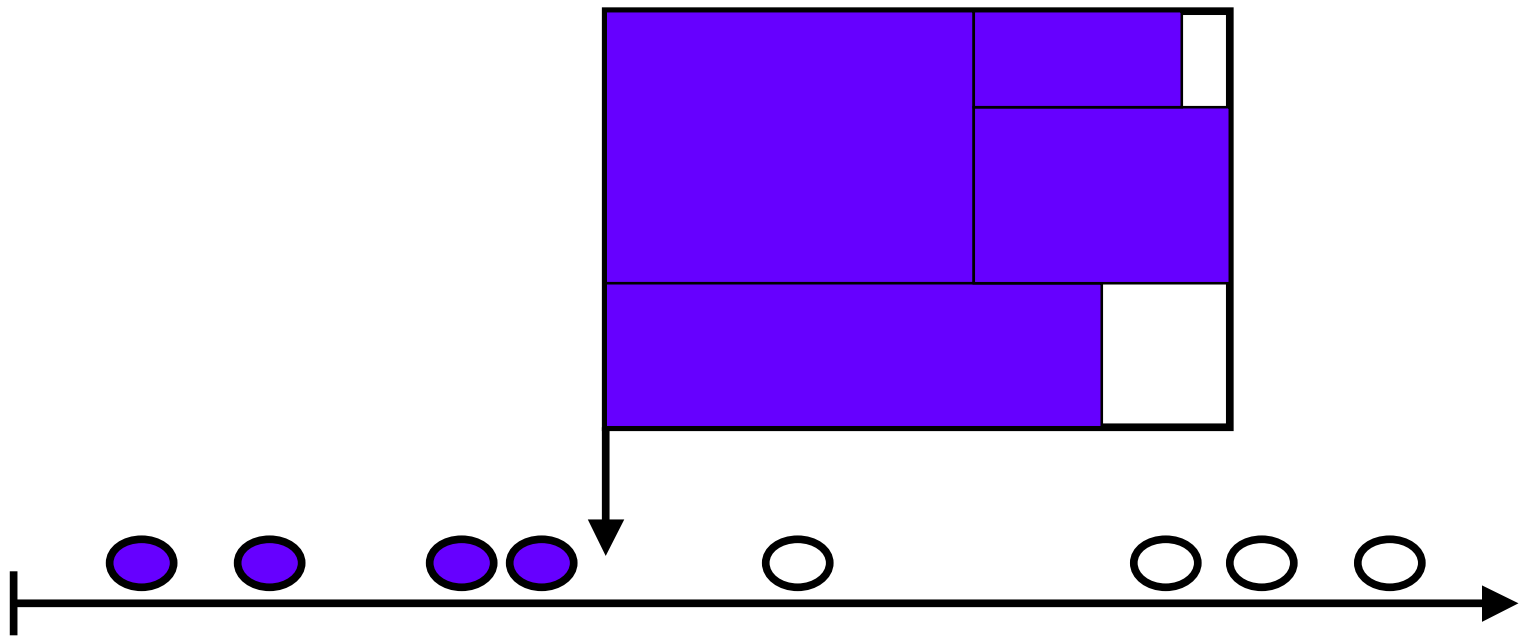
Batch scheduling

Principle: several jobs are treated at once using off-line scheduling.

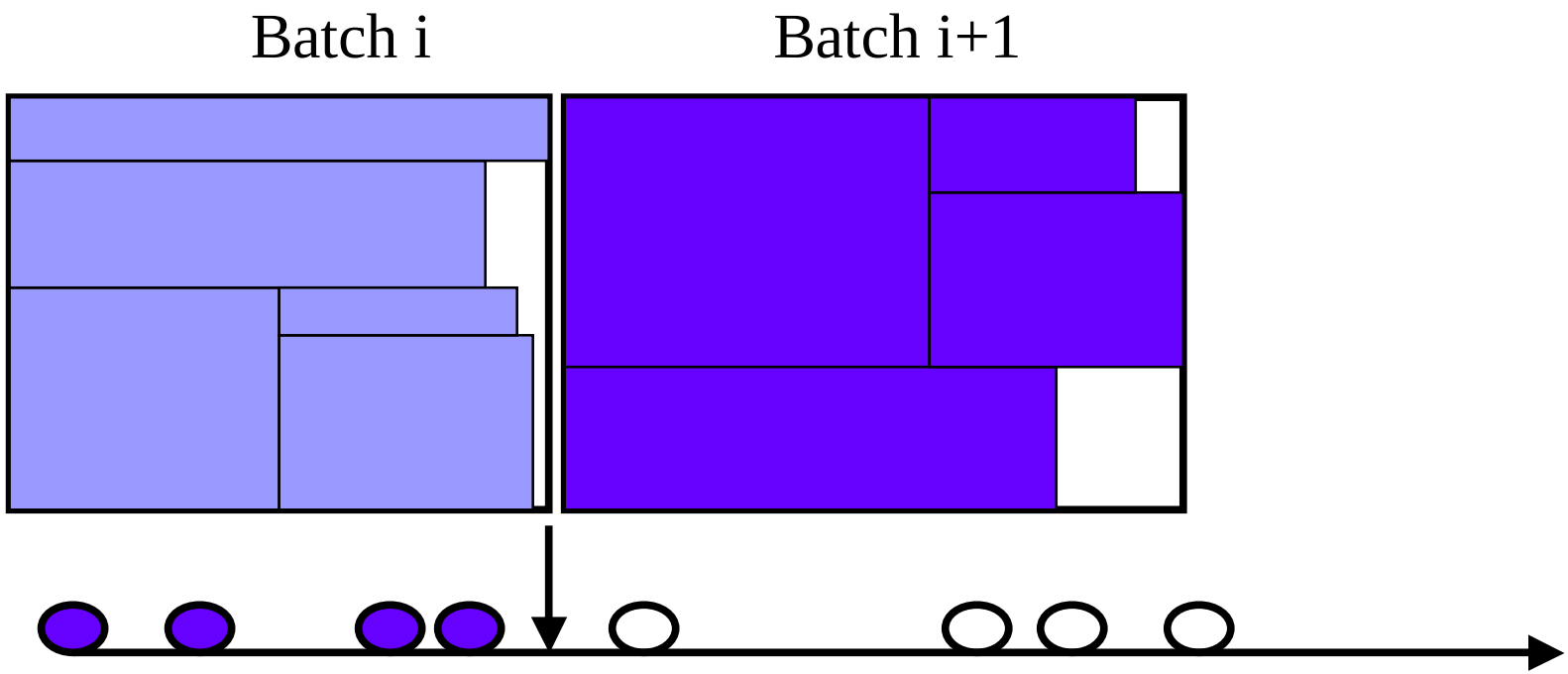
Principle of batch




Start
batch 1



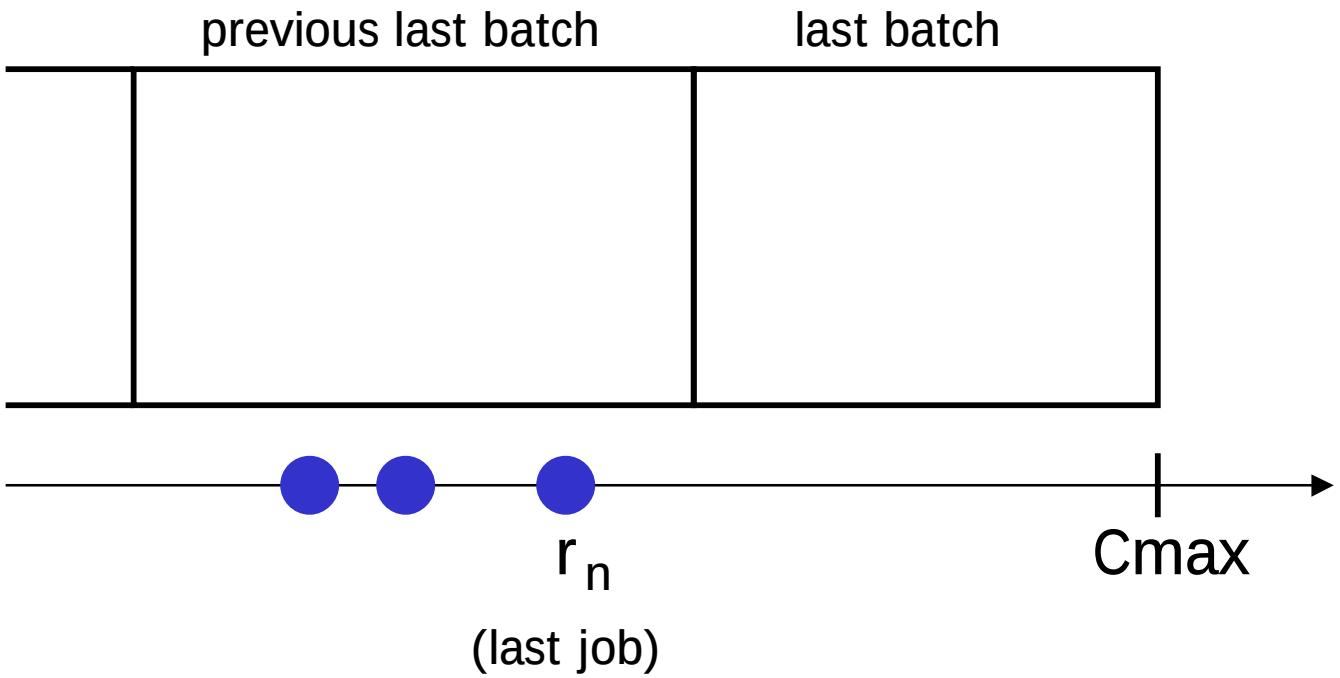
Batch chaining

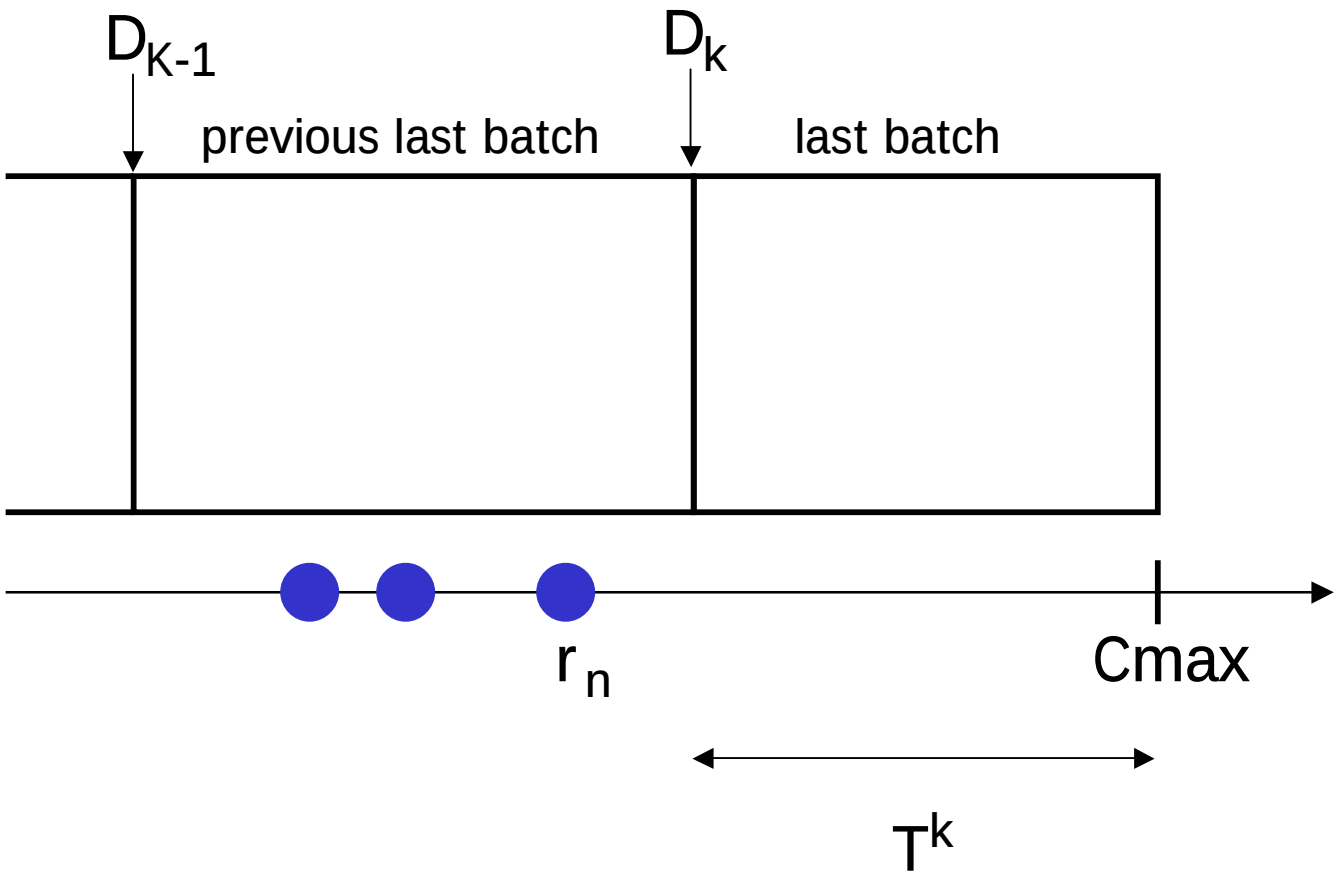


Constructing a batch scheduling

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

Analysis [Shmoys]




D_{K-1}
D_k
previous last batch
last batch
r_n
Cmax
T^k

Proposition

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

Analysis

Tk is the duration of the last batch

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

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

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

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

Application

Applied to the previous 3/2-approximation algorithm, we obtain a 3-approximation on-line batch algorithm for Cmax.

Multi criteria

The Makespan is not always the adequate criterion.

User point of view:

Average completion time (weighted or not)

Other criteria: Stretch, Asymptotic throughput

How to deal with this problem?

Hierachy: one after the other

(Convex) combination of criteria

Transform one criterion in a constraint

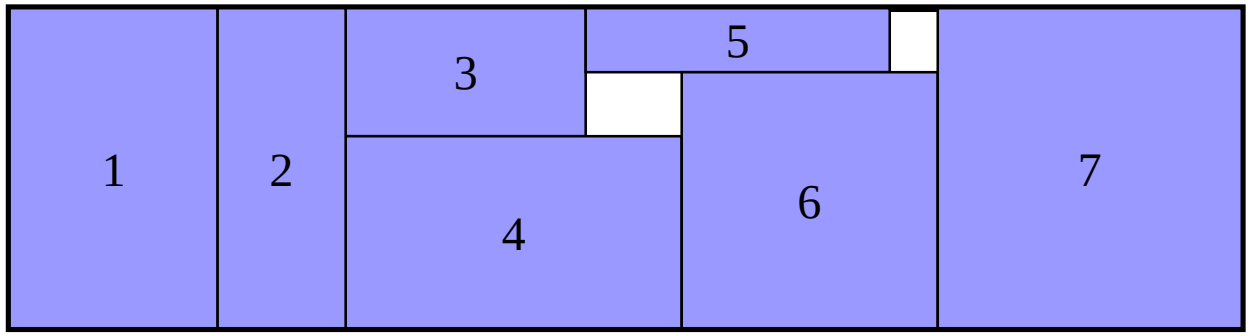
Ad hoc algorithms

A first solution

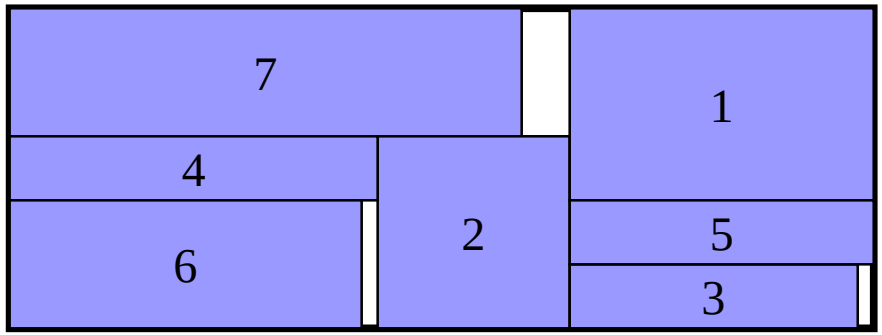
Construct a feasible schedule from two schedules of guaranty r for minsum and r' for makespan with a guaranty (2r,2r') [Stein et al.].

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

Schedules s and s’

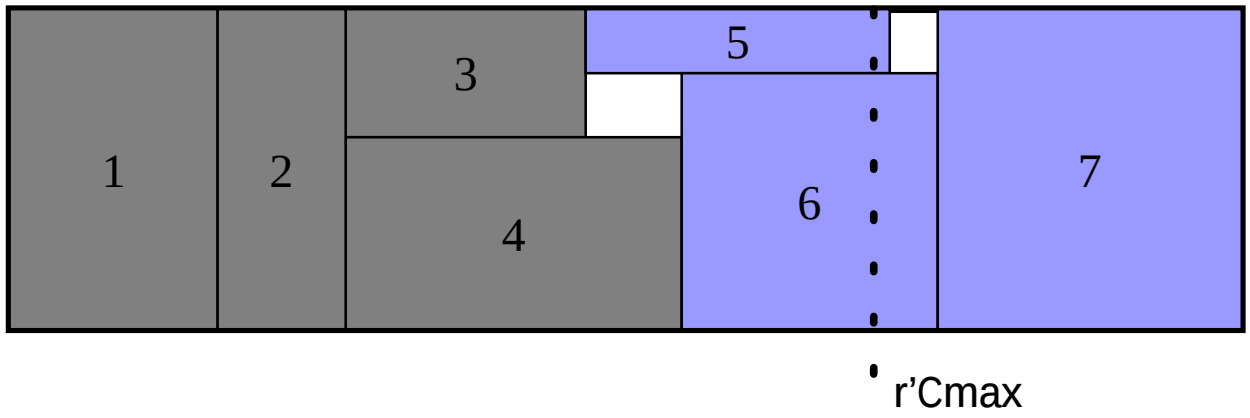


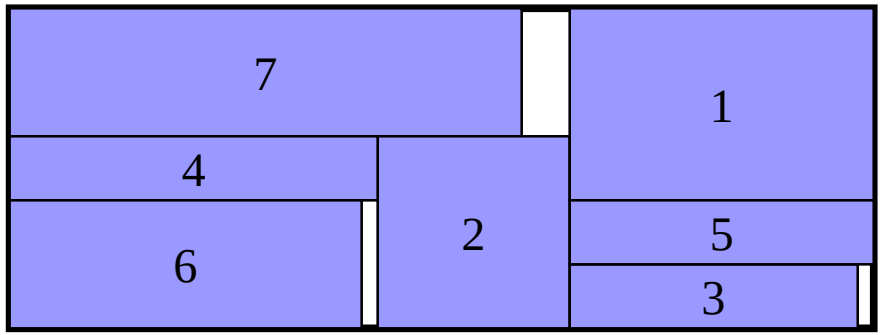
Schedule s
(minsum)



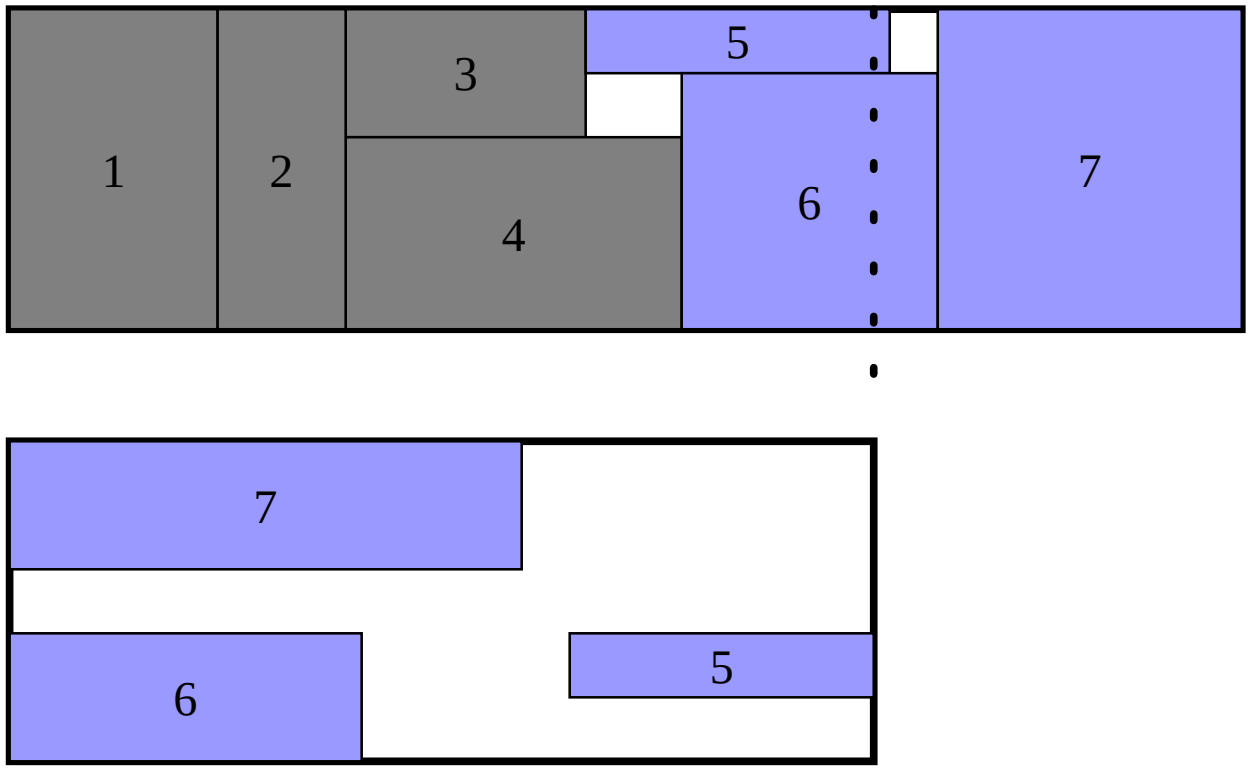
Schedule s’
(makespan)

New schedule

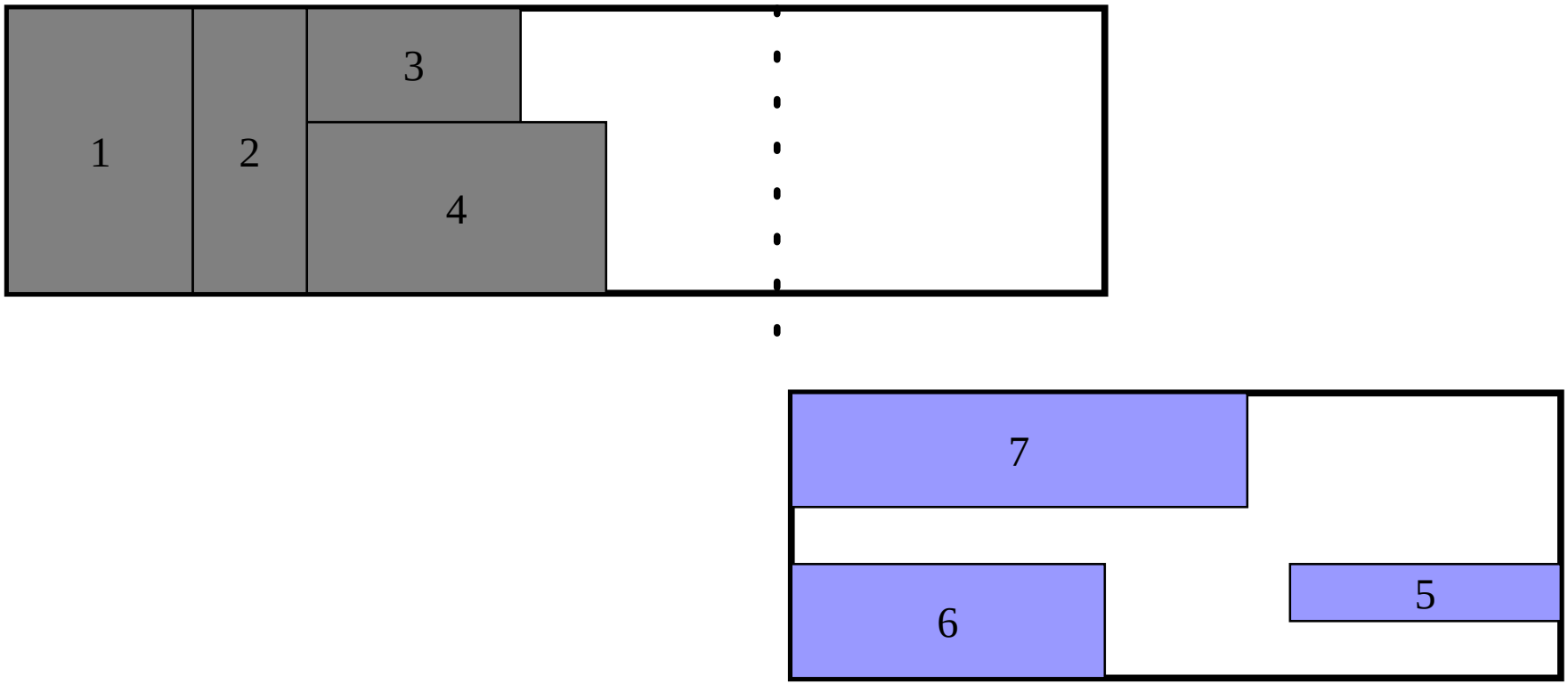




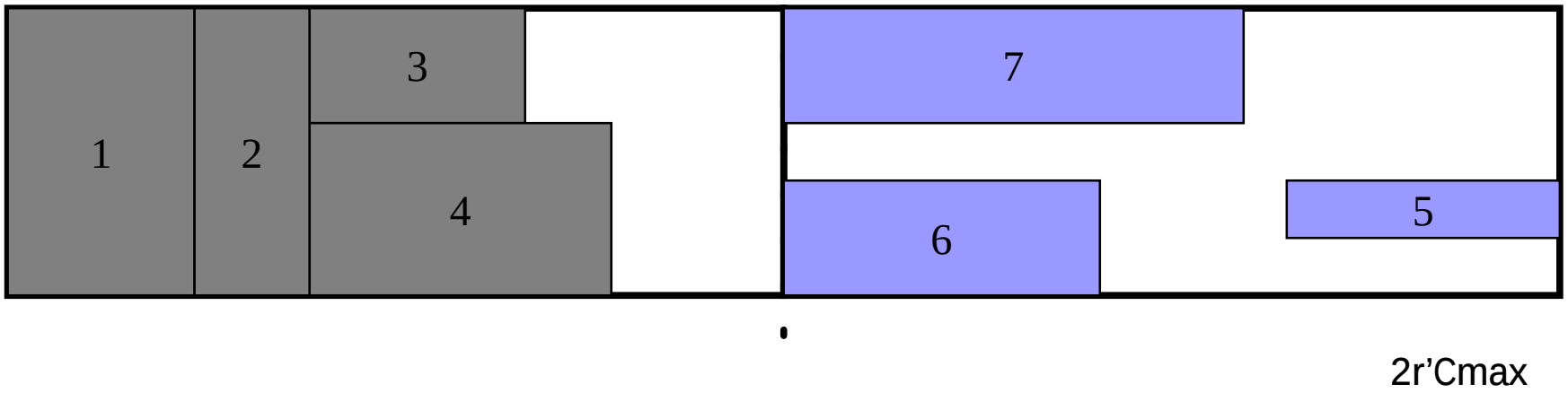
New schedule



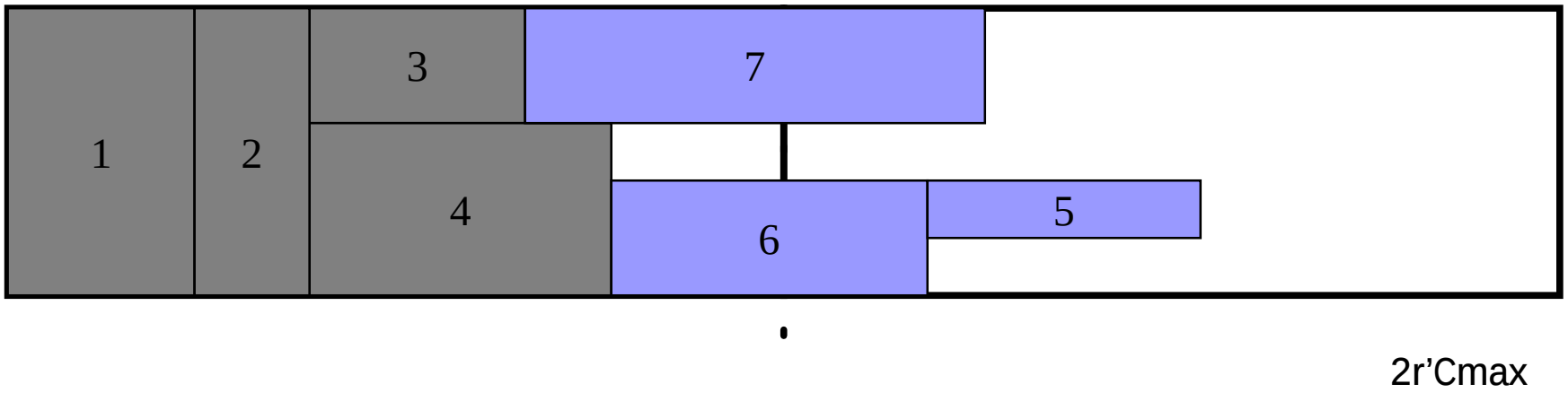
New schedule



New schedule



New schedule



Similar bound for the first criterion

Analysis

The best known schedules are:
8 [Schwiegelsohn] for minsum and 3/2 [Mounie et al.] for makespan leading to (16;3).

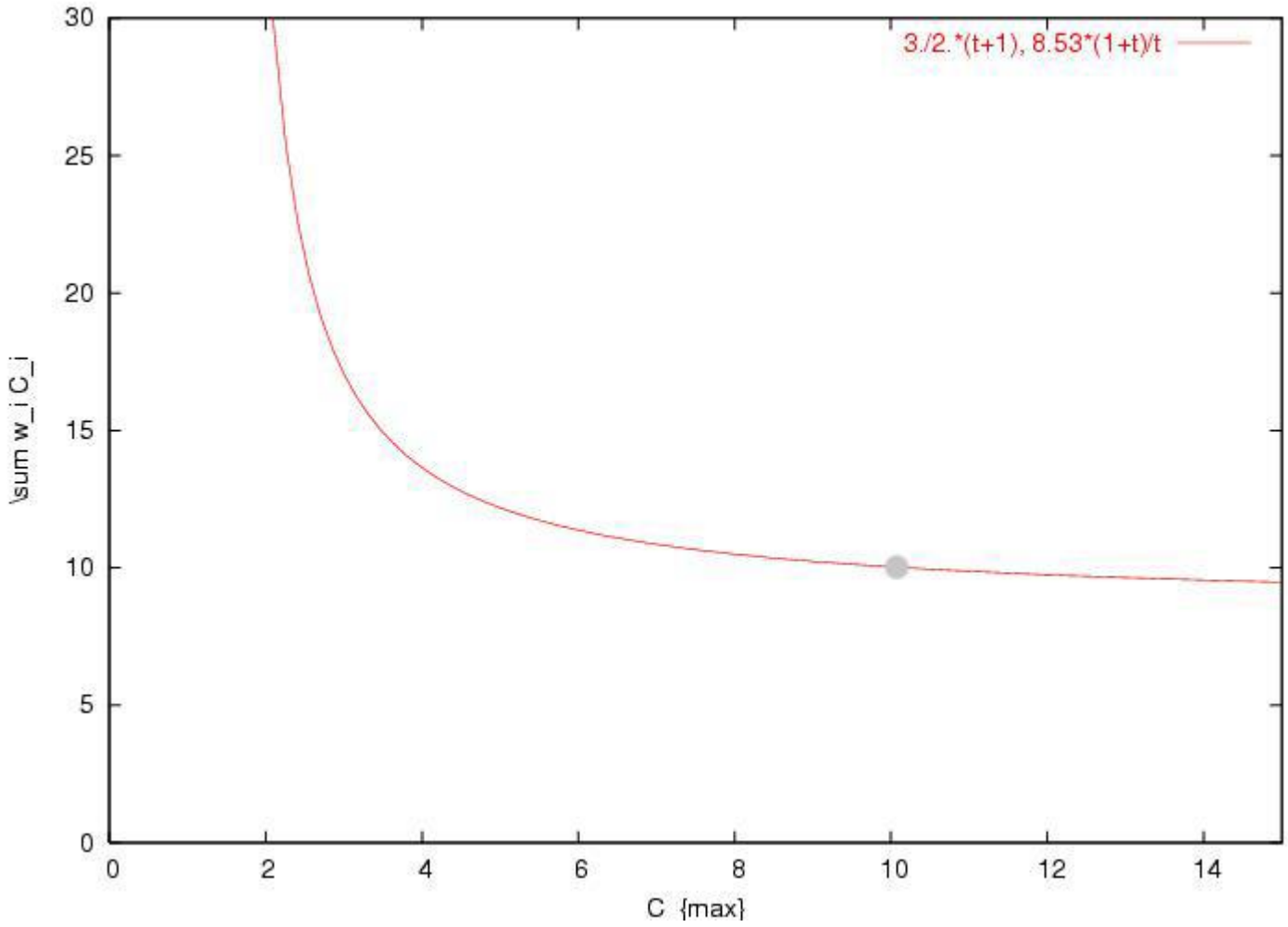
Similarly for the weighted minsum (ratio 8.53).

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 λ r'Cmax

Pareto curve



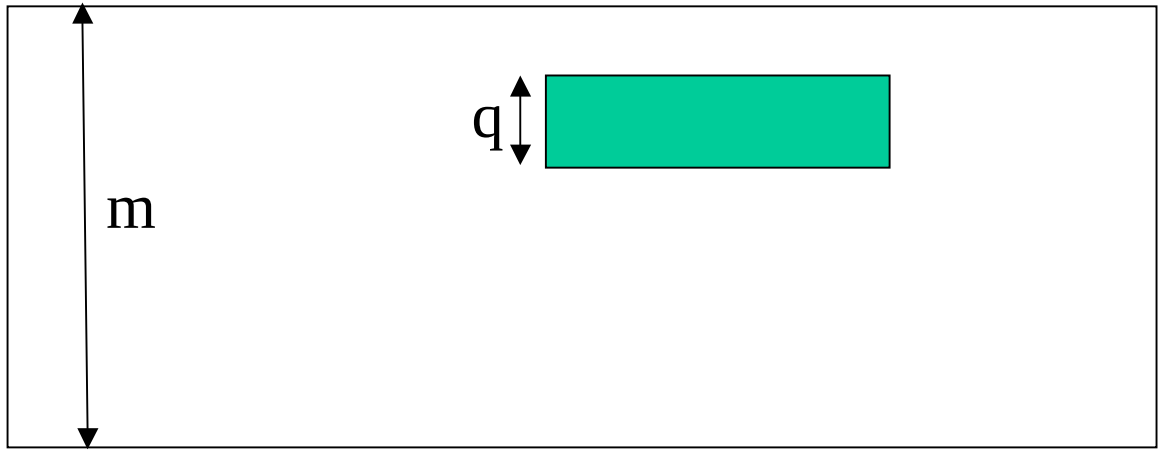
Another way for designing better schedules

We proposed a new solution for a better bound which has not to consider explicitly the schedule for minsum (based on a dynamic framework).

Principle: recursive doubling with smart selection (using a knapsack) inside of each interval.
Starting from the previous algorithm for Cmax, we obtain a (6;6) approximation.

Reservations

On-going work: packing with constraints



Inter-Cluster Scheduling

Context

Load-balancing problem

independent jobs (moldable – non rigid - tasks) are submitted to local schedulers at any time.

Each cluster has its own managing internal rule.

How to manage?

Need of alternative approaches

Too many parameters...

Game theory (prisonner dilema)

Analogy with economy (auctions)

To open the discussion

New execution supports are more and more complex, need of multi-criteria results. Ex: quality of service, reliability, etc..

Need of robust approaches, able to remain efficient if the instances are disturbed...