Approximation Algorithms for Scheduling: some simple examples

Imed Kacem

kacem@univ-metz.fr

LCOMS, Université de Lorraine

http://kacem.imed.perso.neuf.fr/site/

EJC 2015, 4 septembre 2015, Metz, France

UNIVERSITÉ DE LORRAINE

LCOMS

Outline

- Some problems in industrial systems and new technologies
- Common structure
- Complexity and diversity
- Used approaches (exact and heuristic)
- Approximation techniques
- Conclusions

Approximation Techniques

Polynomial approximation

« It is the ***art*** to achieve, in polynomial time, feasible solutions with objective value as close as possible (in some predefined sense) to the optimal » Vangelis Paschos

The motivations:

- Practical motivation:
  - In several situations, we need to achieve feasible solutions in a reasonable time (polynomial complexity).
  - The quality of a solution is generally very important.

- Theoretical motivation:
  - Polynomial approximation and combinatorial optimization are strongly related and the knowledge in one field of them can significantly contribute to the other.
  - Polynomial approximation can be used to evaluate and to study the complexity of discrete optimization problems.

Polynomial approximation: Notations

- I denotes an instance of a discrete optimization problem π (minimization problem).

- OPT(I) is the value of an optimal solution for I

- H is an algorithm for solving π
- H(I) is the value of the solution produced by H for I

- r(H) is the standard approximation ratio defined as the maximum of H(I)/OPT(I) over I

- The closer the ratio r(H) to 1, the better the performance of H

Polynomial approximation: Classes

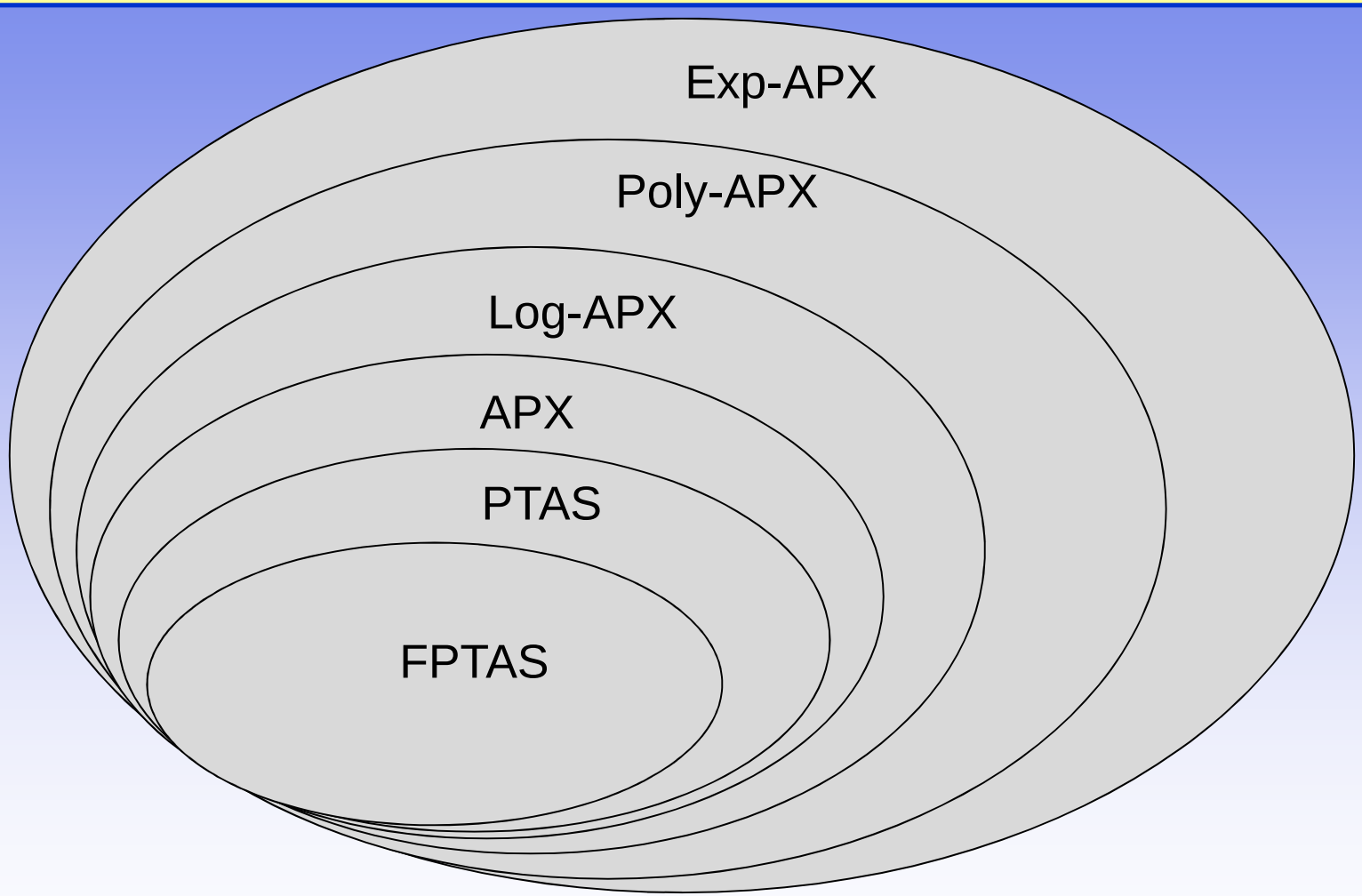
- ***Ratios depending on*** the size of *I* (| *I* |)
  - **Exp-APX** (Travelling Salesman Problem)
  - **Poly-APX** (Graph Coloring Problem)
  - **Log-APX** (Set Covering Problem)

- ***Constant ratios (independent of*** |*I*|***)***
  - **APX** (R||Cmax)

- ***Ratios*** 1 + **ε*, for any*** **ε** > 0:
  - *Polynomial time approximation scheme* (complexity polynomial in |*I*| but, eventually, exponential in 1/**ε**)
    PTAS (Pm||Cmax)

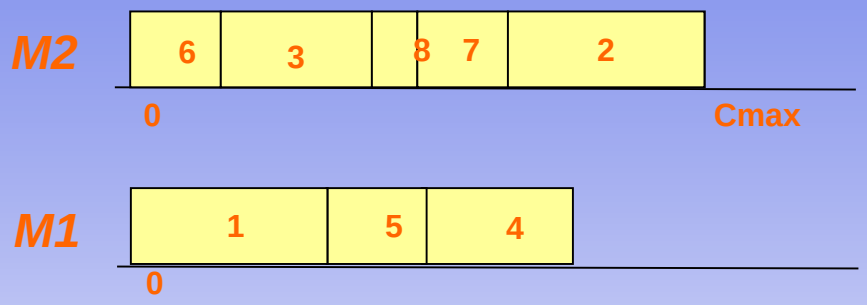
  - *Fully polynomial time approximation scheme* (polynomial in both |*I*| and 1/**ε**)
    FPTAS (2||Cmax)

Polynomial approximation: Classes



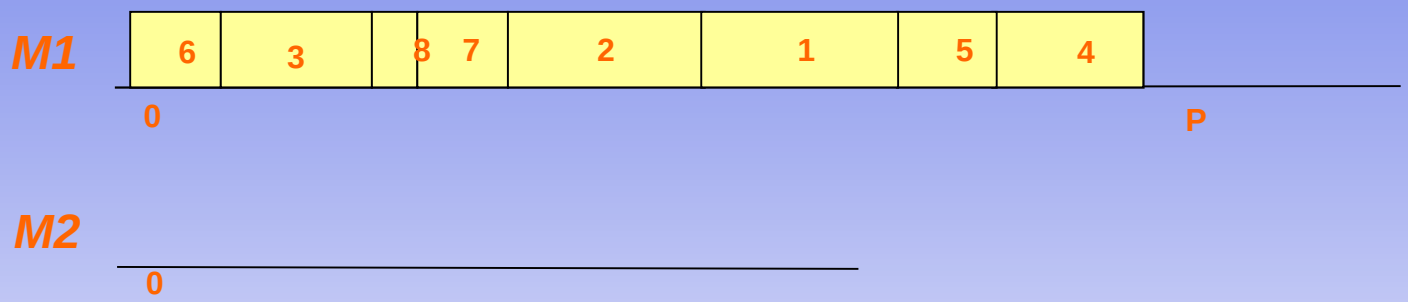
Approximability classes for NP-hard problems (*under the assumption* P ≠ NP)
[V. Paschos, livre chez Hermès]

Polynomial Approximation: Illustrations on 2||Cmax



- The problem is to schedule n jobs on two parallel identical machines, with the aim of minimizing the makespan (Cmax).
- Every job i has a processing time pi.
- The machine is available at time 0 and can process at most one job at a time.
- Without loss of generality, we consider that all the data are integers and that jobs are sorted in the LPT order : $p1 \geq p2 \geq \ldots \geq pn$.
- An optimal solution is composed of two sequences of jobs assigned to the machines. In the two sequences any order is optimal. P is the total processing time.
- The problem is NP-hard in the ordinary sense.

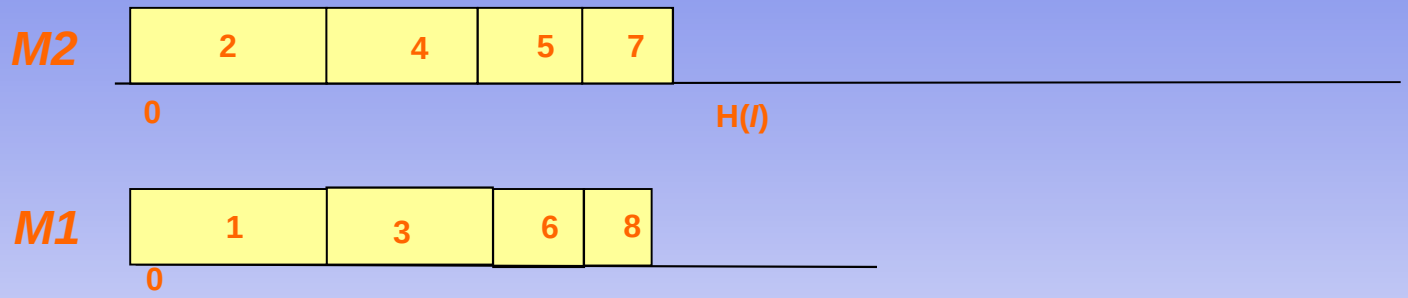
Constant Approximation: Illustrations on 2||Cmax



PROPERTY:
Any assignment of jobs to the two machines is a constant approximation with a standard ratio no more than 2.

Proof:
For any instance I, we have $OPT(I) \geq P/2$ and $H(I) \leq P$.
Then, $H(I) \leq P \leq 2.OPT(I)$.

Constant Approximation: A better heuristic H for 2||Cmax

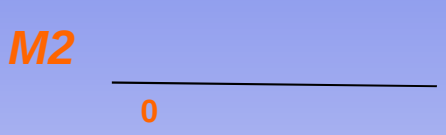


Heuristic description:
Assign the jobs in the LPT order as soon as possible to the available machine.

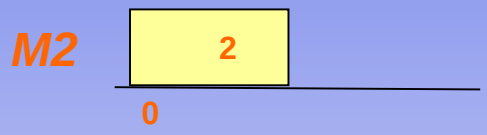
Standard approximation ratio:
For any instance I, we have H(I) ≤ 7/6.OPT(I).

Sketch of proof:
H is optimal for n=1, 2, 3 and 4.

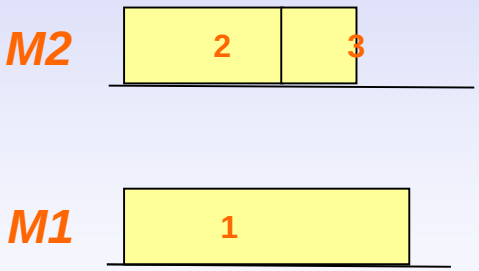
Constant Approximation: A better heuristic H for 2||Cmax



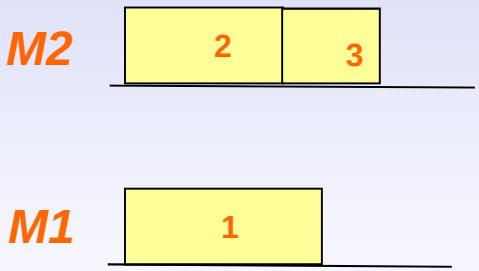
H is optimal for n=1.



H is optimal for n=2.

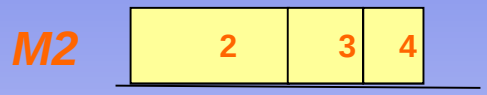


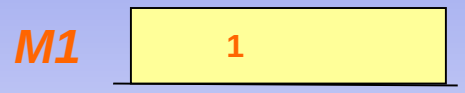
H is optimal for n=3
(case 1: p1>p2+p3).



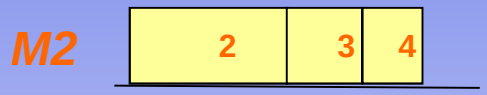
H is optimal for n=3
(case 2: p1 ≤ p2+p3).

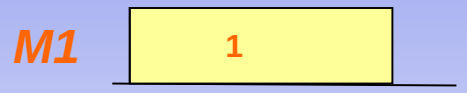
Constant Approximation: A better heuristic H for 2||Cmax



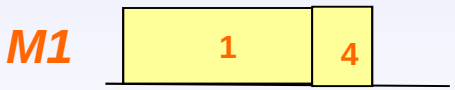


H is optimal for n=4
(case 1: p1>p2+p3+p4).

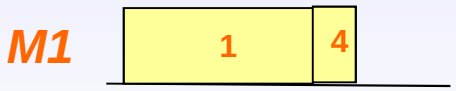




H is optimal for n=4
(case 2: p2+p3 ≤ p1 < p2+p3+p4).

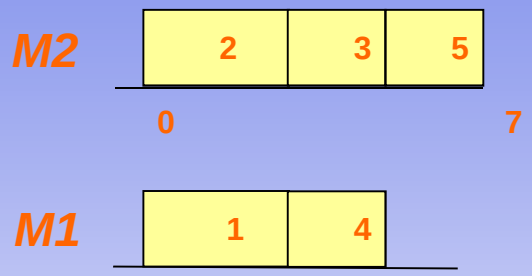


H is optimal for n=4
(case 3: p1 < p2+p3).

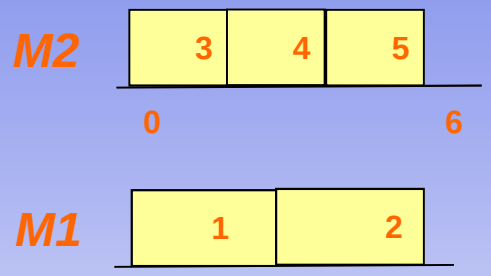


H is optimal for n=4
(case 4: p1 < p2+p3).

Constant Approximation: A better heuristic H for 2||Cmax



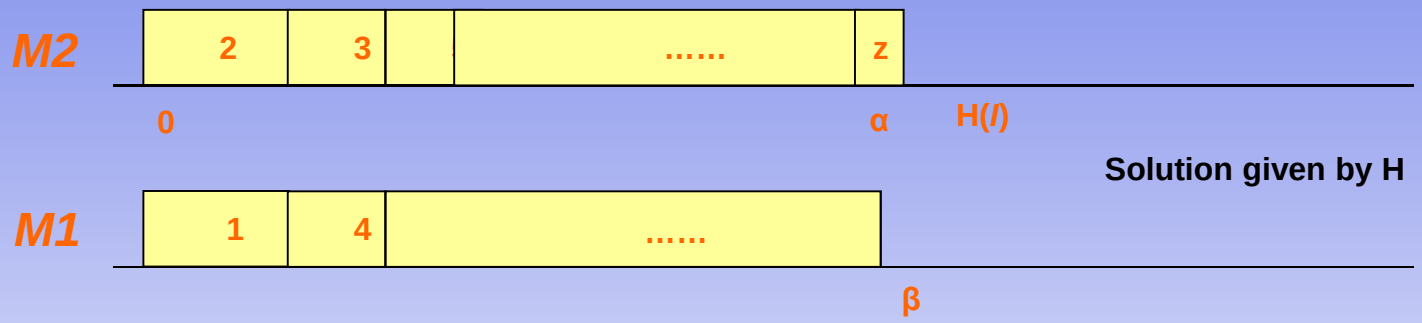
Solution given by H



Optimal solution

H is not optimal for n=5
p1 = 3
p2 = 3
p3 = 2
p4 = 2
p5 = 2

Constant Approximation: A better heuristic H for 2||Cmax



Let z denote the last job scheduled in this sequence. Let **α** be the starting time of *z* and let **β** be the completion time on the other machine. We have:

H(I)= α + pz

α ≤ β

OPT(I) ≥ P/2 = (α + pz + β)/2

Hence,

H(I) – OPT(I) ≤ α + pz - (α + pz + β)/2 = (α + pz - β)/2 ≤ pz/2

Moreover, **OPT(I) ≥** $\lceil z/2 \rceil$**.pz** and **z** can be considered **≥ 5.**

Then, **(H(I) – OPT(I))/OPT(I) ≤ 1/(2**$\lceil z/2 \rceil$**) ≤ 1/6 = 0.16667** which implies that:

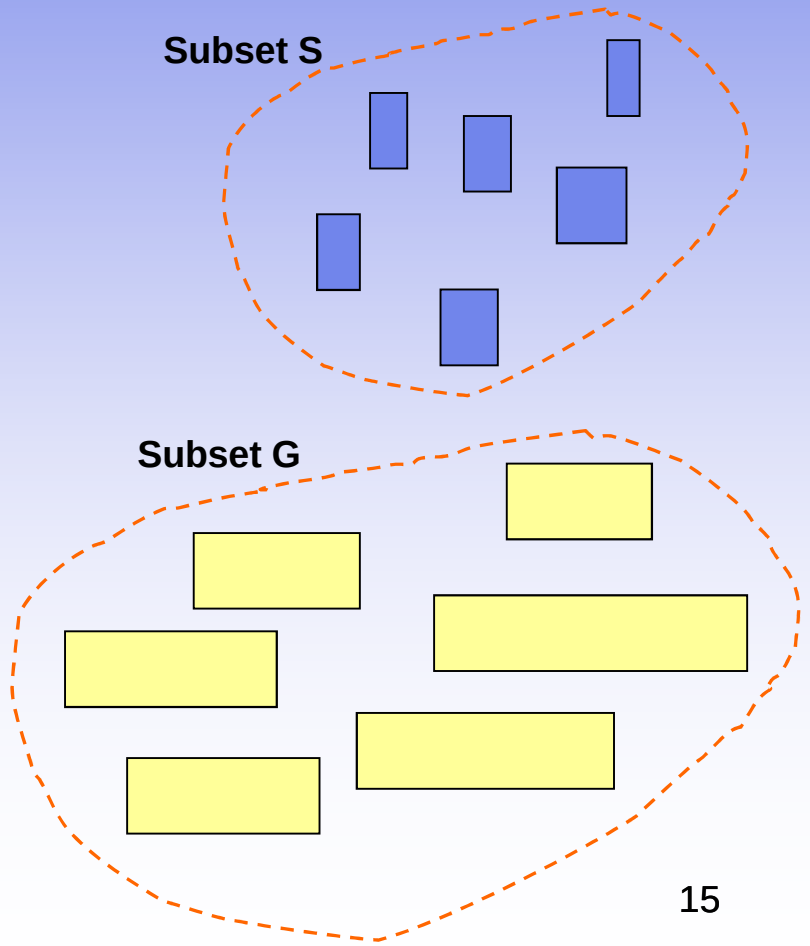
H(I) ≤ 1.16667 OPT(I).

PTAS exists:

A better algorithm than H for 2||Cmax

The idea is very simple!
Let $\varepsilon > 0$.

1. **We divide the set of jobs in two subsets G and S where G = { i | pi > εP/2} and S = { i | pi ≤ εP/2}.**
2. **Enumerate all the assignments of G on the two machines (their cardinal is limited to $2^{2/\varepsilon}$).**
3. **For every assignment A generated in Step (2) complete by the jobs of S by scheduling them iteratively on the less loaded machine (in any order).**
4. **Select the best schedule from the solutions obtained in Step (3).**



PTAS exists:

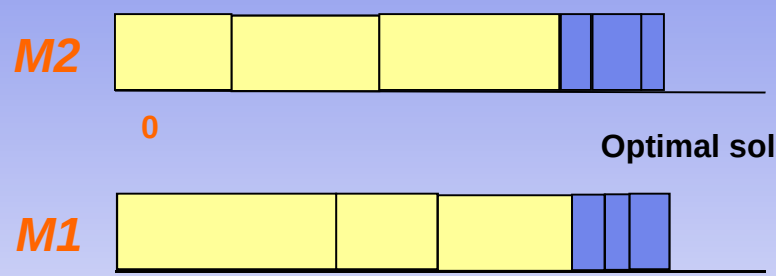
A better heuristic algorithm than H for 2||Cmax

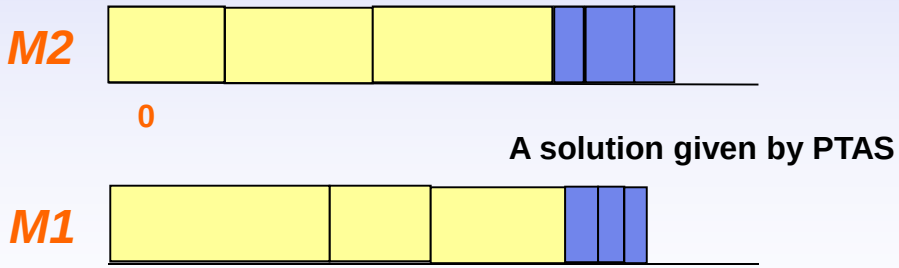
For any optimal solution OPT there exists a solution generated by PTAS where we have the same assignment of great jobs (from G).

Only the assignments of S may be different.

The difference cannot be more than the size of a small job $\varepsilon P/2 \leq \varepsilon OPT$.

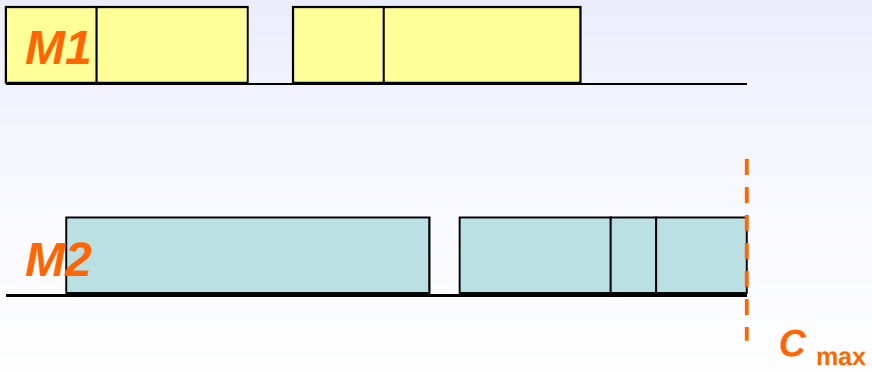
The complexity is bounded by $O(n.2^{1/\varepsilon})$.





FPTAS: An illustration for 2|rj|Cmax

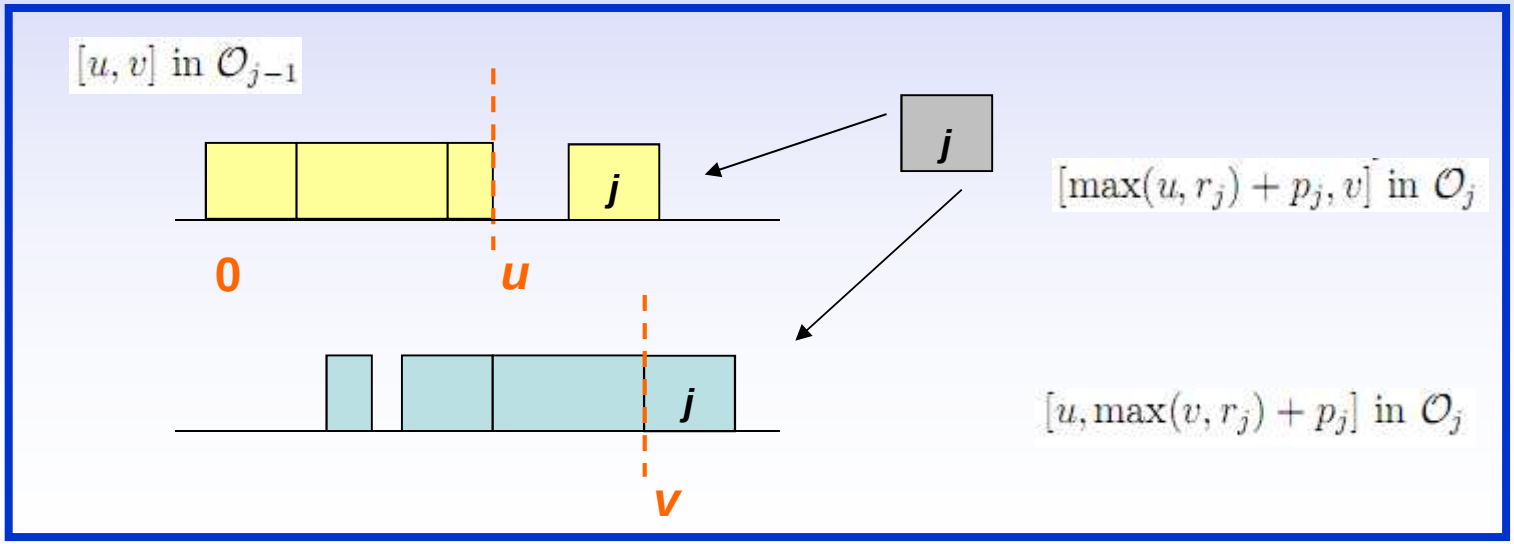
- The problem is to schedule n jobs on two parallel identical machines, with the aim of minimizing the makespan (Cmax).
- Every job i has a processing time pi and a release date ri.
- The machine is available at time 0 and can process at most one job at a time.
-Without loss of generality, we consider that all the data are integers and that jobs are sorted in the FIFO order : $r1 \leq r2 \leq \ldots \leq rn$.
- An optimal solution is composed of two FIFO sequences of jobs assigned to the machines. In the two sequences only the FIFO order is optimal. The problem is NP-hard in the ordinary sense.



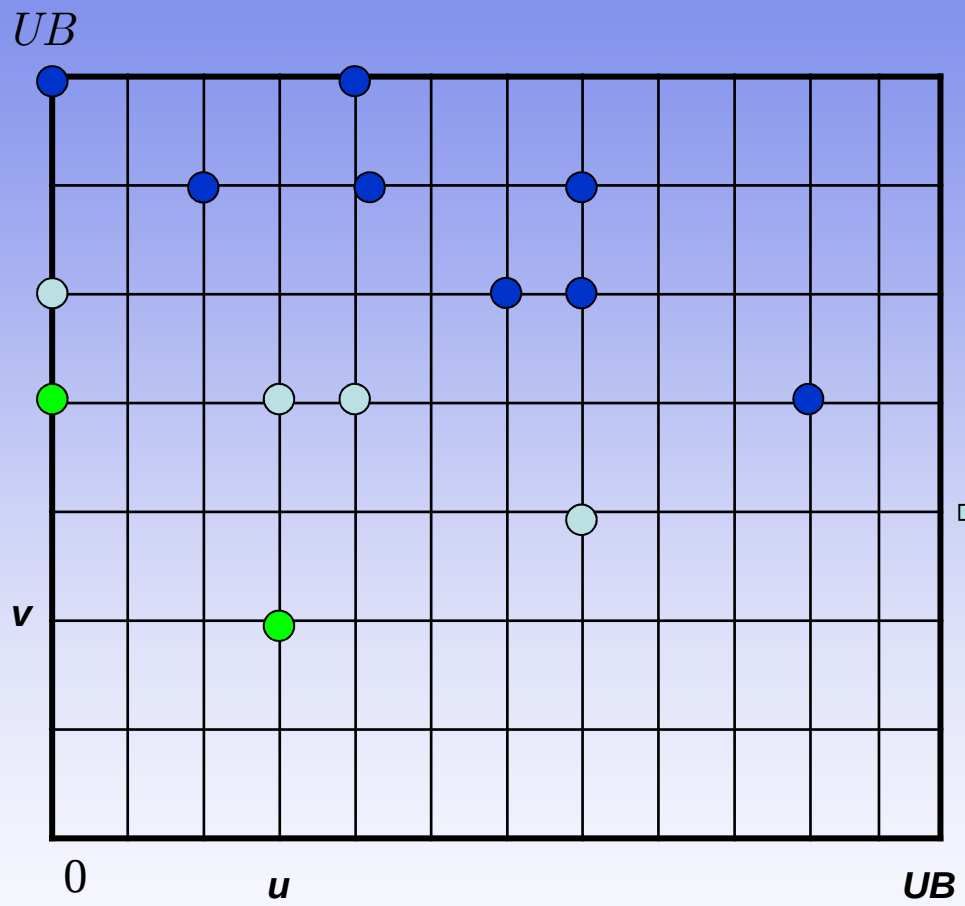
17

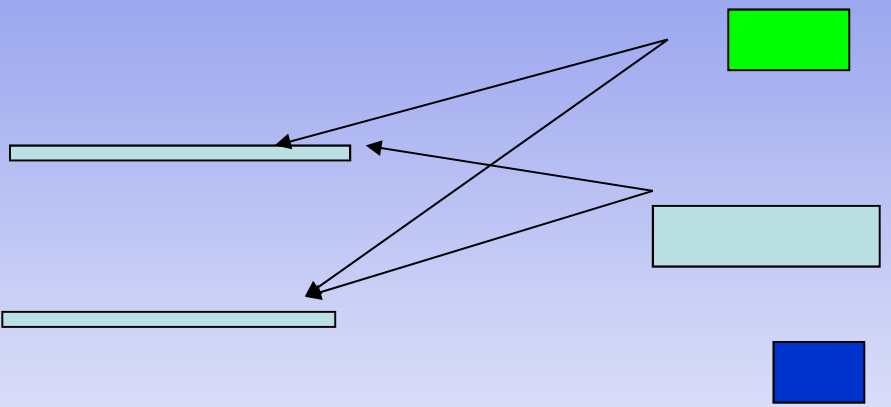
Dynamic Programming

The problem can be optimally solved by applying the following standard dynamic programming algorithm A. This algorithm generates iteratively some sets of states. At every iteration j, a set $\mathcal{O}_j$ composed of states is generated $(1 \leq j \leq n)$. Each state $[u, v]$ in $\mathcal{O}_j$ can be associated with a feasible schedule for the first j jobs. Variable u denotes the completion time of the last job scheduled on the first machine and v is the completion time of the last job scheduled on the second machine. This algorithm can be described as follows:



New Dynamic Programming: Illustration





Complexity of $O\left(n\overline{C_{\max}}\right)$

Fully Polynomial Time Approximation Scheme (FPTAS)

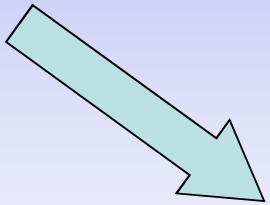
$$LB = \frac{2C^{H'}_{\max}(\mathcal{P})}{3},$$

$$\beta = \left\lceil \frac{3n}{2\varepsilon} \right\rceil,$$

$$\gamma = \frac{C^{H'}_{\max}(\mathcal{P})}{\beta}.$$

DEFINITION: Given ε>0, an FPTAS finds (1+ ε)-approximation with a time-complexity polynomial in (1/ ε) and in the input size.

PRINCIPLE: modification of the execution of an exact algorithm

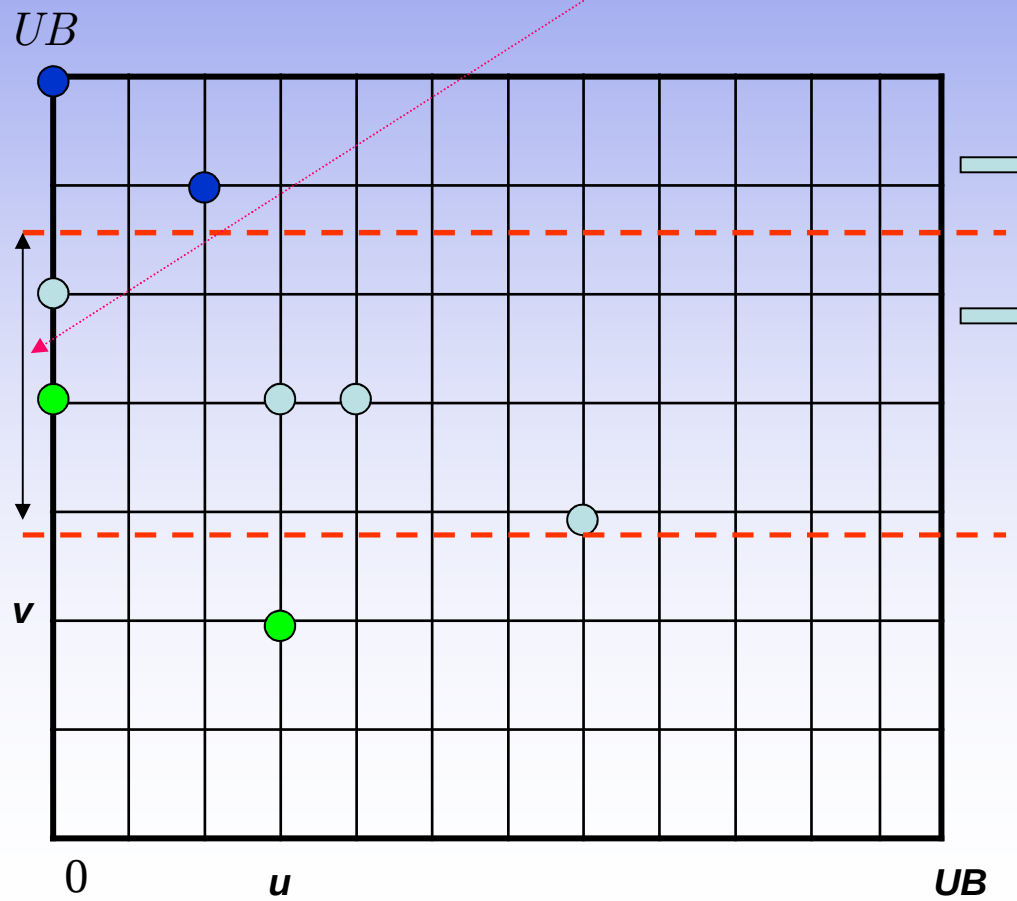


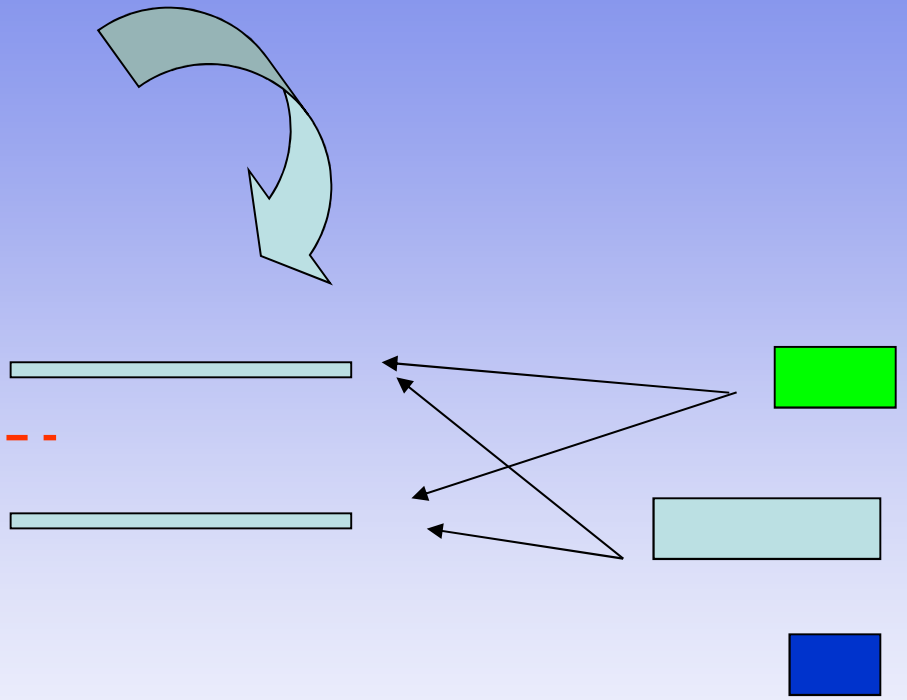
We split the interval $[0, \overline{C_{\max}}]$ into β equal subintervals $I_h = [(h-1)\gamma, h\gamma]_{1 \le h \le \beta}$ of length γ. Our algorithm $\mathcal{A}_\varepsilon$ generates reduced sets $\mathcal{O}_j^{\#}$ instead of sets $\mathcal{O}_j$. It can be described as follows:

$$LB = \frac{2C^{H'}_{\max}(\mathcal{P})}{3},$$

$$\beta = \left\lceil \frac{3n}{2\varepsilon} \right\rceil,$$

$$\gamma = \frac{C^{H'}_{\max}(\mathcal{P})}{\beta}.$$





Complexité de A'ε : $O(n^2/\varepsilon)$

Fully Polynomial Time Approximation Scheme (FPTAS): the main results [Kacem 2009]

Lemma 3 *For every state* $[u, v]$ *in* $\mathcal{O}_j$ *there exists a state* $\left[u^{\#}, v^{\#}\right]$ *in* $\mathcal{O}_j^{\#}$ *such that:*

$$u^{\#} - u \leq 0 \tag{7}$$

and

$$v^{\#} - v \leq j\gamma \tag{8}$$

Theorem 1 *Given an arbitrary* $\varepsilon > 0$*, algorithm* $\mathcal{A}_\varepsilon$ *yields an output* $C_{\max}^{\mathcal{A}_\varepsilon}(\mathcal{P})$ *such that:*

$$\frac{C_{\max}^{\mathcal{A}_\varepsilon}(\mathcal{P})}{C_{\max}^{*}(\mathcal{P})} \leq (1+\varepsilon). \tag{9}$$

Lemma 4 *Given an arbitrary* $\varepsilon > 0$*, algorithm* $A(\varepsilon)$ *can be implemented in* $O(n^2/\varepsilon)$ *time.*

Some results in approximation theory

Topics	Surveys
Bin Packing	L. HALL
Covering and Packing	D. HOCHBAUM
Scheduling	D. SCHMOYS
Knapsack	H. KELLERER
Symmetric Quadratic Knapsack	I. KACEM, H. KELLERER, V. STRUSEVICH
Graph Coloring	V. PASCHOS

Thank you!