On-line Scheduling Algorithms

Resources

- As far as the exam is concerned, it is enough studying (and then deeply understanding) the following slides.

- However, if interested, a good book about on-line computation is the following:

  A. Borodin, R. El-yaniv. Online computation and Competitive analysis. Cambridge University Press.

Introduction.

- In all previous slides about scheduling the underlying assumptions were based on the fact that all the problem data (e.g., number of jobs, processing times, weights, and so on) are known in advance.
- The (schedule) decision-maker can determine at time zero the entire schedule while having all the information at his disposal.
- This most common paradigm is usually referred to as offline scheduling.

Introduction. (2)

- One category of machine scheduling problems are the so-called *online scheduling* problems.
- In an online scheduling problem the decision-maker does not know in advance how many jobs have to be processed and what the processing times are.
- The decision-maker becomes aware of the existence of a job only when the job is released and presented to him.
- The decision-maker only has knowledge on machines.

Introduction. (3)

- On-line scheduling can be seen as decision making with incomplete information.

- At certain points, decisions have to be made without knowing the complete instance.

An online model: Jobs one by one

- In such a scenario we have initial knowledge on machines.
- Suppose that jobs are ordered in some list (sequence).
- Jobs are presented one by one to the decision maker.
- The moment the job is presented, its characteristics get available.
- The decision for an item has to be taken before the next item is presented.
- The decision is irrevocable.

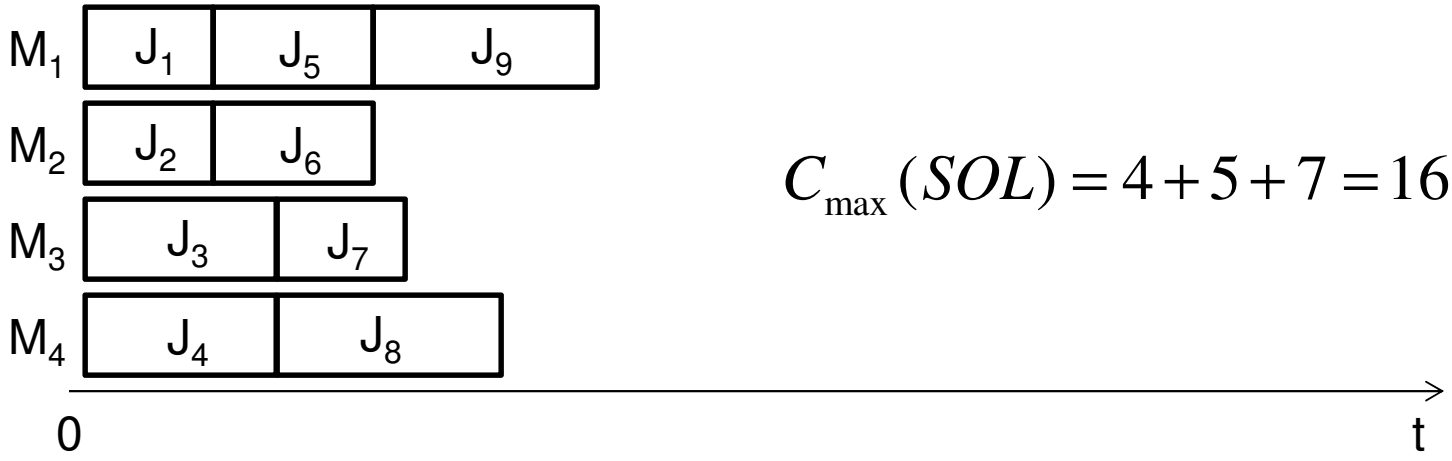
Jobs one by one: an example.

- Suppose we have 4 identical machines.
- Jobs are presented one by one.
- We have to schedule jobs on machines with the objective of minimizing the machine MAKESPAN.
- A natural approach (similar to the one used in Algorithm MAKESPAN) could be assigning the next job to the machine with minimum load (or minimum machine completion time).
- However, notice that here we do not assign jobs ordered by their processing time, since we do not know jobs in advance.
- As usual, we denote by *SOL* the solution (schedule) returned by our algorithm and by *OPT* an optimal solution to the problem.
- Moreover let $C_{max}(SOL)$ (respectively $C_{max}(OPT)$) be the Machine MAKESPAN of the schedule *SOL* (respectively of the optimal schedule *OPT*).

Jobs one by one: an example. (2)

- **Jobs List**: jobs appear in the following order:

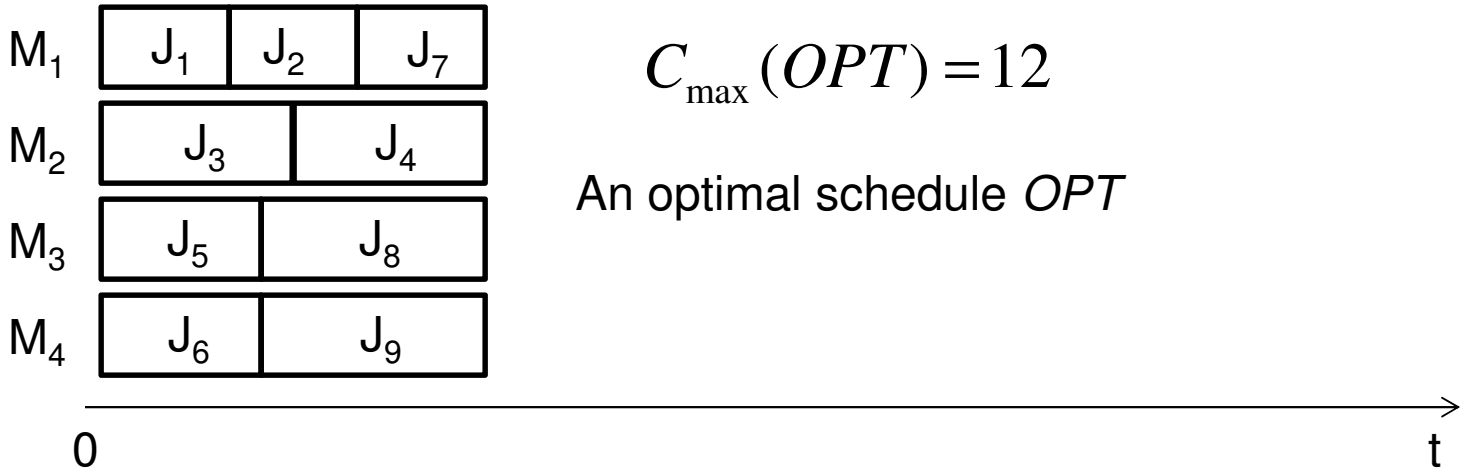
p_1=4 p_2=4 p_3=6 p_4=6 p_5=5 p_6=5 p_7=4 p_8=7 p_9=7



$$C_{\max}(SOL) = 4 + 5 + 7 = 16$$

Jobs one by one: an example. (3)

p_1=4; p_2=4; p_3=6; p_4=6; p_5=5; p_6=5; p_7=4; p_8=7; p_9=7.



$$C_{\max}(OPT) = 12$$

An optimal schedule *OPT*

We have the following trivial lower bound of an optimal solution:

$$C_{\max}(OPT) \geq \frac{\sum_{j=1}^{n} p_j}{m}$$

- In fact in this case $\frac{\sum_{j=1}^{n} p_j}{m} = \frac{4+4+6+6+5+5+4+7+7}{4} = \frac{48}{4} = 12$

Performance measure

- Quality of an on-line algorithm is measured by evaluating its worst case performance (like performance analysis we have seen for the (approximation Algorithm MAKESPAN) .

- As reference value the best off-line value is used.

- We are going to define performance measure while dealing with the objective of minimizing the machine MAKESPAN.

Performance measure – Competitive analysis.

- An on-line algorithm is *c-competitive* if its objective value is no more than *c* times the optimal off-line value for all instances.
- Formally, let *SOL* be the solution (schedule) returned by our on-line algorithm and let *OPT* be an optimal off-line solution to the problem.
- Moreover let $C_{max}(SOL)$ (respectively $C_{max}(OPT)$) be the Machine MAKESPAN of the schedule *SOL* (respectively of the optimal schedule *OPT*).

Performance measure – Competitive analysis. (2)

- We say that an on-line algorithm is *c-competitive* (where $c \geq 1$), if for any possible input sequence:

$$\frac{C_{\max}(SOL)}{C_{\max}(OPT)} \leq c$$

Performance measure – Complexity analysis.

- For the on-line scenario we do not focus on the computational complexity analysis.

- However the algorithm we are going to analyze is really simple.

Online Scheduling Identical Machine MAKESPAN (minimization) problem.

- INPUT: *m identical* machine (*h = 1, . . .,m*), *Jobs* ***???*** .
- OUTPUT: a schedule ***S=****(*$S_{1,\ldots,}S_m$*).*
- GOAL*:* Minimizing the Machine MAKESPAN, that is minimizing

$$Max_{h=1,\ldots,m}\{MC_h(S)\} = C_{\max}(S)$$

- Remind that:
  - Here jobs are presented one by one to the decision maker.
  - The moment the job is presented, its characteristics get available.
  - The decision for an item has to be taken before the next item is presented.
  - The decision is irrevocable.

Online Scheduling Identical Machine MAKESPAN (minimization) problem: a simple algorithm.

DEFINITION: let $T_h(j)$ be the machine completion time of the machine *h* at the time where *j* jobs have been allocated to machines.

ALGORITHM LIST:

1. j=0;
2. Whenever a new job arrives:

   - j=j+1;

   - assign the new job to a machine with minimum $T_h(j-1)$; *(if there are more than one machines with minimum* $T_h(j-1)$ *choose any one of them).*

Algorithm LIST: performance analysis.

Theorem 6: *Algorithm LIST is a* $2-\frac{1}{m}$ *competitive algorithm for the Online Scheduling Identical Machine MAKESPAN (minimization) problem.*

Proof of Theorem 6.

Proof:

- Let *SOL* be the solution (schedule) returned by our algorithm LIST and let *OPT* be an optimal solution to the problem.
- Consider the job that finishes as the last one in *SOL*. Let us call such job *l* and denote by p_l its processing time. Clearly, the machine that schedules job *l* is the last one to finish and therefore it gives the machine MAKESPAN of the solution *SOL*.
- Suppose that job *l* was started at time *t*.
- Since job *l* has been assigned to the machine with minimum completion processing time, it follows that at all times before *t*, all the machines are busy, as otherwise the last job (i.e., the job *l*) would be scheduled earlier.

- Hence $\mathrm{C}_{\max}(OPT) \geq t + \frac{p_l}{m}$, as the optimal schedule has to schedule all the jobs, and the sum of processing times of all the jobs is at least $\mathrm{m}t + p_l$ (recall the lower bound to an optimal solution)

Proof of Theorem 6. (2)

- Moreover it is easy to see that $C_{max}(OPT) \geq p_l$, as the optimal schedule takes time p_l to process even this single job.

- Combining the last two inequalities we get that the Machine MAKESPAN of the online solution returned by our Algorithm LIST which is $\boldsymbol{t + p_l}$ is bounded by:

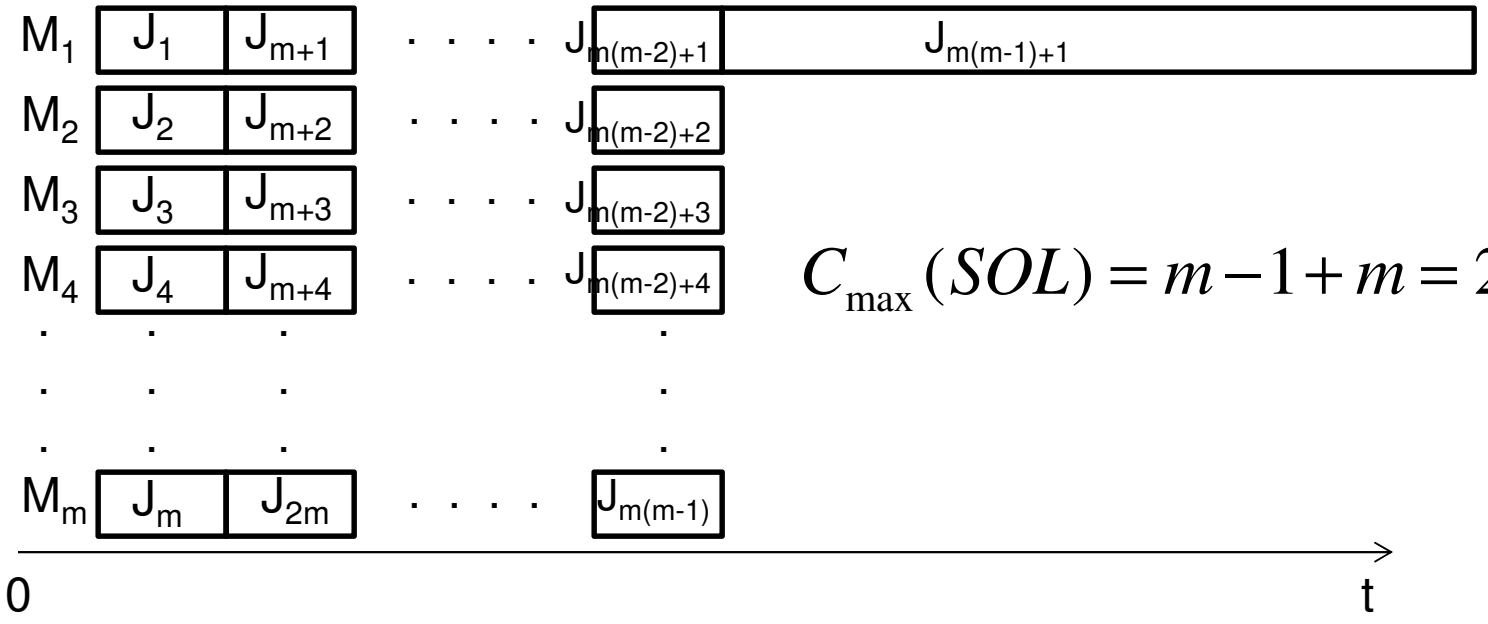
$$t + p_l = \underbrace{t + \frac{p_l}{m}} + (1 - \frac{1}{m})\underbrace{p_l} \leq (2 - \frac{1}{m})C_{\max}(OPT)$$

$$C_{\max}(OPT) \geq t + \frac{p_l}{m} \qquad C_{\max}(OPT) \geq p_l$$

□

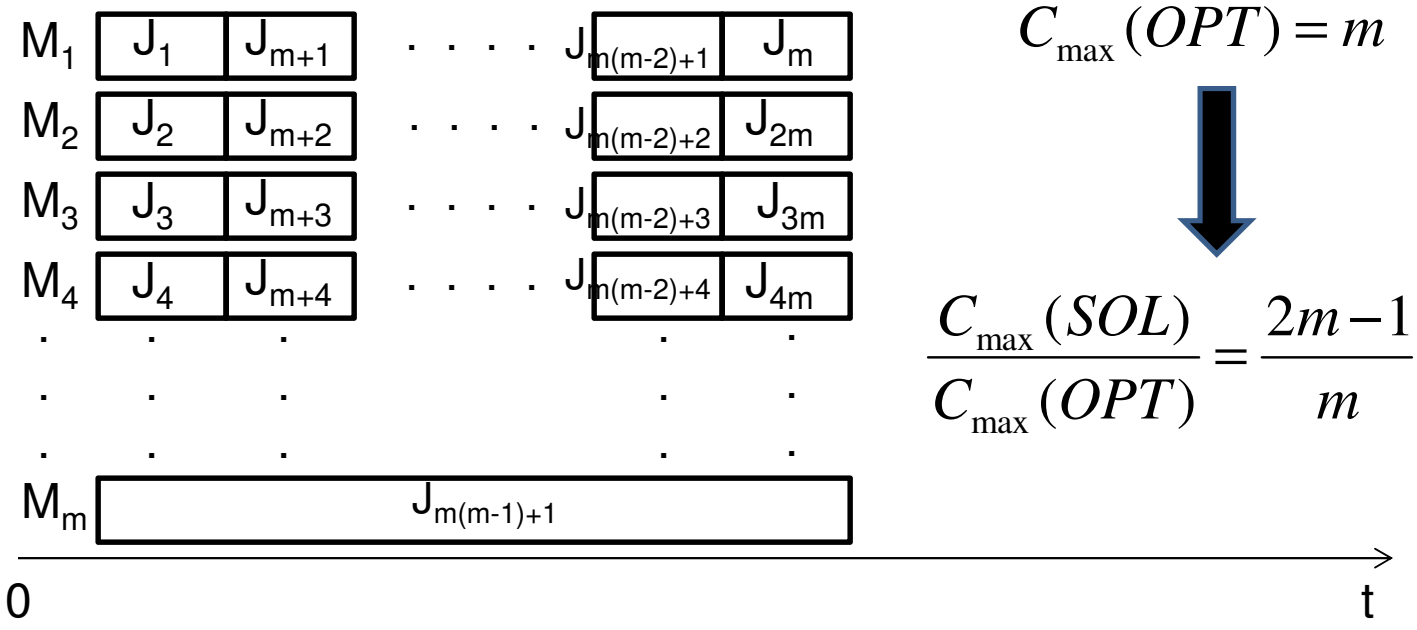
A worst case example of Algorithm LIST.

- Could we prove in general a strictly better competitive ratio (i.e., smaller than *2 - 1/m*) for the algorithm LIST when applied to the Online Scheduling Identical Machine MAKESPAN (minimization) problem?
- Unfortunately not! As proved by the following example.
- Consider the sequence of *m(m-1)* jobs with processing time *1* followed by one job with processing time *m*.



$$C_{\max}(SOL) = m-1+m = 2m-1$$

A worst case example of Algorithm LIST. (2)



$$C_{\max}(OPT) = m$$

$$\frac{C_{\max}(SOL)}{C_{\max}(OPT)} = \frac{2m-1}{m} = 2 - \frac{1}{m}$$