

Scheduling Jobs on Identical Parallel Machines

Given n jobs, where job $j \in \{1, \dots, n\}$ has processing time p_j .
Schedule the jobs on m identical parallel machines such that the **Makespan** (finishing time of the last job) is minimized.

$$\begin{array}{ll} \min & L \\ \text{s.t.} & \forall \text{ machines } i \quad \sum_j p_j \cdot x_{j,i} \leq L \\ & \forall \text{ jobs } j \quad \sum_i x_{j,i} \geq 1 \\ & \forall i, j \quad x_{j,i} \in \{0, 1\} \end{array}$$

Here the variable $x_{j,i}$ is the decision variable that describes whether job j is assigned to machine i .

Lower Bounds on the Solution

Let for a given schedule C_j denote the finishing time of machine j , and let C_{\max} be the makespan.

Let C_{\max}^* denote the makespan of an optimal solution.

Clearly

$$C_{\max}^* \geq \max_j p_j$$

as the longest job needs to be scheduled somewhere.

Lower Bounds on the Solution

The average work performed by a machine is $\frac{1}{m} \sum_j p_j$.
Therefore,

$$C_{\max}^* \geq \frac{1}{m} \sum_j p_j$$

Local Search

A local search algorithm successively makes certain small (cost/profit improving) changes to a solution until it does not find such changes anymore.

It is conceptually very different from a Greedy algorithm as a feasible solution is always maintained.

Sometimes the running time is difficult to prove.

Local Search for Scheduling

Local Search Strategy: Take the job that finishes last in the produced schedule. Move it to another machine. If there is such a move that reduces the makespan, perform the switch.

REPEAT

Local Search Analysis

Let ℓ be the job that finishes last in the produced schedule.

Let S_ℓ be its start time, and let C_ℓ be its completion time.

Note that every machine is busy before time S_ℓ , because otherwise we could move the job ℓ and hence our schedule would not be locally optimal.

We can split the total processing time into two intervals one from 0 to S_ℓ the other from S_ℓ to C_ℓ .

The interval $[S_\ell, C_\ell]$ is of length $p_\ell \leq C_{\max}^*$.

During the first interval $[0, S_\ell]$ all processors are busy, and, hence, the total work performed in this interval is

$$m \cdot S_\ell \leq \sum_{j \neq \ell} p_j.$$

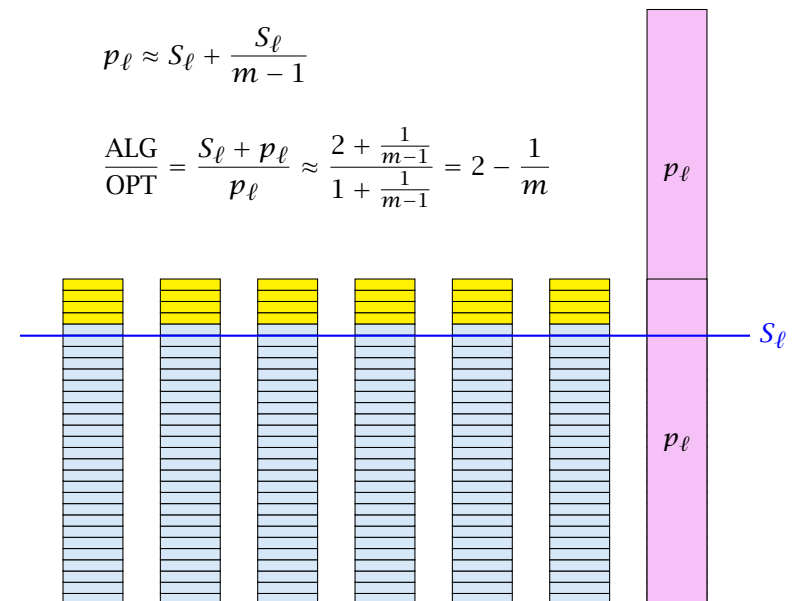
Hence, the length of the schedule is at most

$$p_\ell + \frac{1}{m} \sum_{j \neq \ell} p_j = \left(1 - \frac{1}{m}\right)p_\ell + \frac{1}{m} \sum_j p_j \leq \left(2 - \frac{1}{m}\right)C_{\max}^*$$

A Tight Example

$$p_\ell \approx S_\ell + \frac{S_\ell}{m-1}$$

$$\frac{\text{ALG}}{\text{OPT}} = \frac{S_\ell + p_\ell}{p_\ell} \approx \frac{2 + \frac{1}{m-1}}{1 + \frac{1}{m-1}} = 2 - \frac{1}{m}$$



- ▶ We can assume that one machine schedules p_1 and p_n (the largest and smallest job).
- ▶ If not assume wlog. that p_1 is scheduled on machine A and p_n on machine B .
- ▶ Let p_A and p_B be the other job scheduled on A and B , respectively.
- ▶ $p_1 + p_n \leq p_1 + p_A$ and $p_A + p_B \leq p_1 + p_A$, hence scheduling p_1 and p_n on one machine and p_A and p_B on the other, cannot increase the Makespan.
- ▶ Repeat the above argument for the remaining machines.