Scheduling Jobs on Identical Parallel Machines

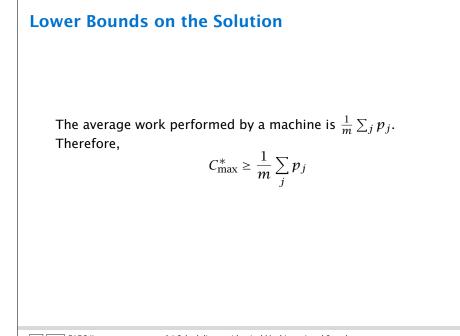
Given *n* jobs, where job $j \in \{1, ..., n\}$ has processing time p_j . Schedule the jobs on m identical parallel machines such that the Makespan (finishing time of the last job) is minimized.

> min L s.t. \forall machines $i \sum_{j} p_j \cdot x_{j,i} \leq L$ $\forall \text{jobs } j \quad \sum_i x_{i,i} \ge 1$ $\forall i, j$ $x_{i,i} \in \{0,1\}$

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

50 00	EADS II © Harald Räcke
	© Harald Räcke

306



Lower Bounds on the Solution

Let for a given schedule C_i 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

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

as the longest job needs to be scheduled somewhere.

EADS II © Harald Räcke

14 Scheduling on Identical Machines: Local Search

Local Search

EADS II

© Harald Räcke

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

It is conceptionally 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 and try to move it to another machine. If there is such a move that reduces the makespan, perform the switch.

REPEAT

EADS II	14 Scheduling on Identical Machines: Local Search	
🛛 🕒 🗋 🕼 © Harald Räcke		310

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

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

EADS II © Harald Räcke

14 Scheduling on Identical Machines: Local Search

312

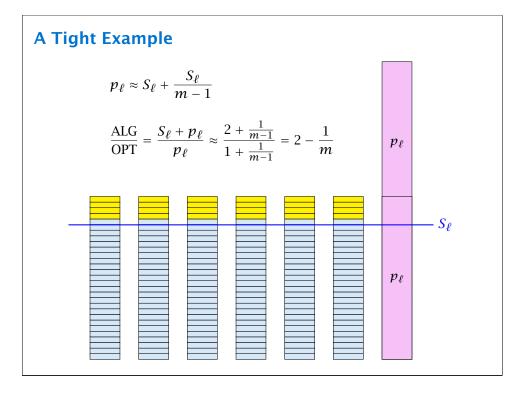
Local Search Analysis

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.

EADS II 14 Scheduling on Identical Machines: Local Search © Harald Räcke 311



A Greedy Strategy

List Scheduling:

Order all processes in a list. When a machine runs empty assign the next yet unprocessed job to it.

Alternatively:

Consider processes in some order. Assign the i-th process to the least loaded machine.

It is easy to see that the result of these greedy strategies fulfill the local optimally condition of our local search algorithm. Hence, these also give 2-approximations.

EADS II © Harald Räcke	15 Scheduling on Identical Machines: Greedy 3	4	\mathbb{N}

Proof:

- Let p₁ ≥ · · · ≥ p_n denote the processing times of a set of jobs that form a counter-example.
- Wlog. the last job to finish is n (otw. deleting this job gives another counter-example with fewer jobs).
- If p_n ≤ C^{*}_{max}/3 the previous analysis gives us a schedule length of at most

$$C_{\max}^* + p_n \le \frac{4}{3} C_{\max}^*$$

Hence, $p_n > C_{\max}^*/3$.

- This means that all jobs must have a processing time $> C_{\text{max}}^*/3$.
- But then any machine in the optimum schedule can handle at most two jobs.
- > For such instances Longest-Processing-Time-First is optimal.

15 Scheduling on Identical Machines: Greedy

A Greedy Strategy

Lemma 2

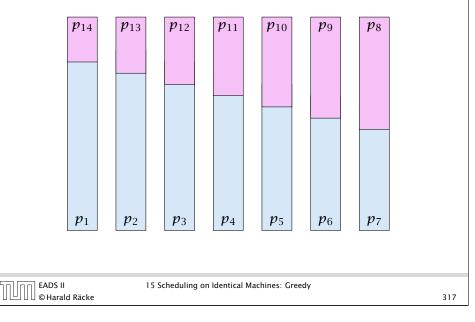
If we order the list according to non-increasing processing times the approximation guarantee of the list scheduling strategy improves to 4/3.

EADS II © Harald Räcke

15 Scheduling on Identical Machines: Greedy

When in an optimal solution a machine can have at most 2 jobs the optimal solution looks as follows.

315



- We can assume that one machine schedules p₁ and p_n (the largest and smallest job).
- If not assume wlog. that p₁ 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.
- $p_1 + p_n \le p_1 + p_A$ and $p_A + p_B \le p_1 + p_A$, 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.

	15 Scheduling on Identical Machines: Greedy	
UUU©Harald Räcke		318



