Introduction 02282

Inge Li Gørtz

Approximation Algorithms

Approximation algorithms

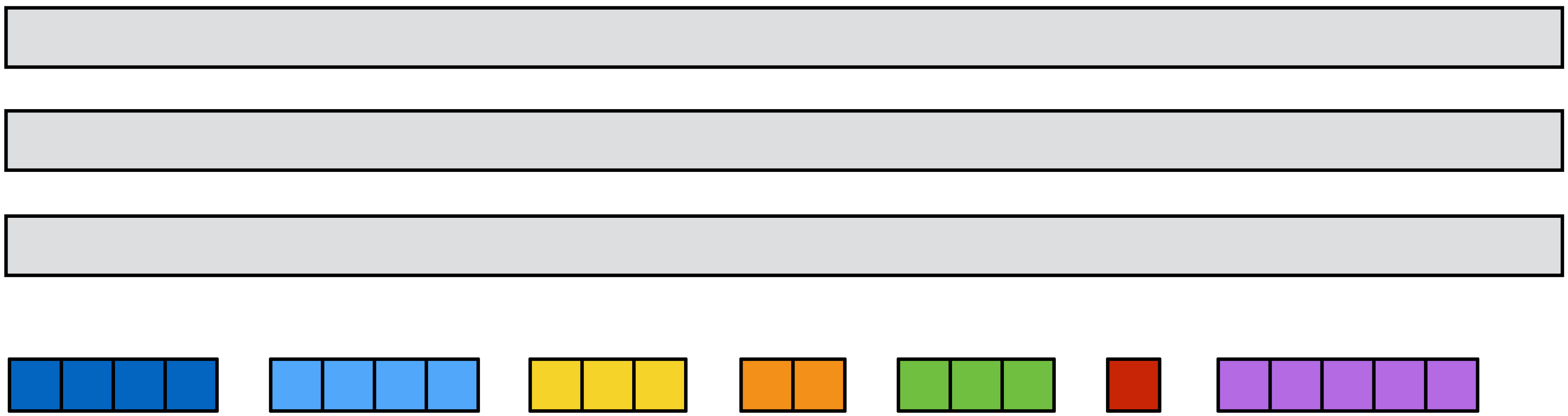
- Fast. Cheap. Reliable. Choose two.
- NP-hard problems: choose 2 of
  - optimal
  - polynomial time
  - all instances

- **Approximation algorithms.** Trade-off between time and quality.

- Let A(I) denote the value returned by algorithm A on instance I. Algorithm A is an *α-approximation algorithm* if for any instance I of the optimization problem:
  - A runs in polynomial time
  - A returns a valid solution
  - $A(I) \leq \alpha \cdot OPT$, where $\alpha \geq 1$, for minimization problems
  - $A(I) \geq \alpha \cdot OPT$, where $\alpha \leq 1$, for maximization problems

Load balancing

Simple greedy (list scheduling)



- *Simple greedy.* Process jobs in any order. Assign next job on list to machine with smallest current load.

- The local search algorithm above is a 2-approximation algorithm:
  - polynomial time ✓
  - valid solution ✓
  - factor 2

Approximation factor

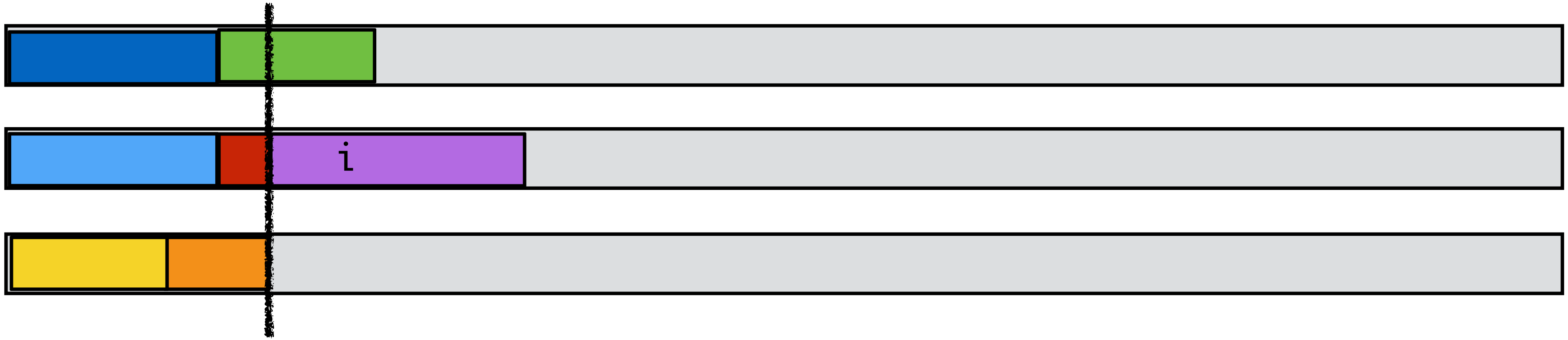
- Lower bounds:
  - Each job must be processed:

$$T^* \geq \max_j t_j$$

  - There is a machine that is assigned at least average load:

$$T^* \geq \frac{1}{m} \sum_j t_j$$

Approximation factor

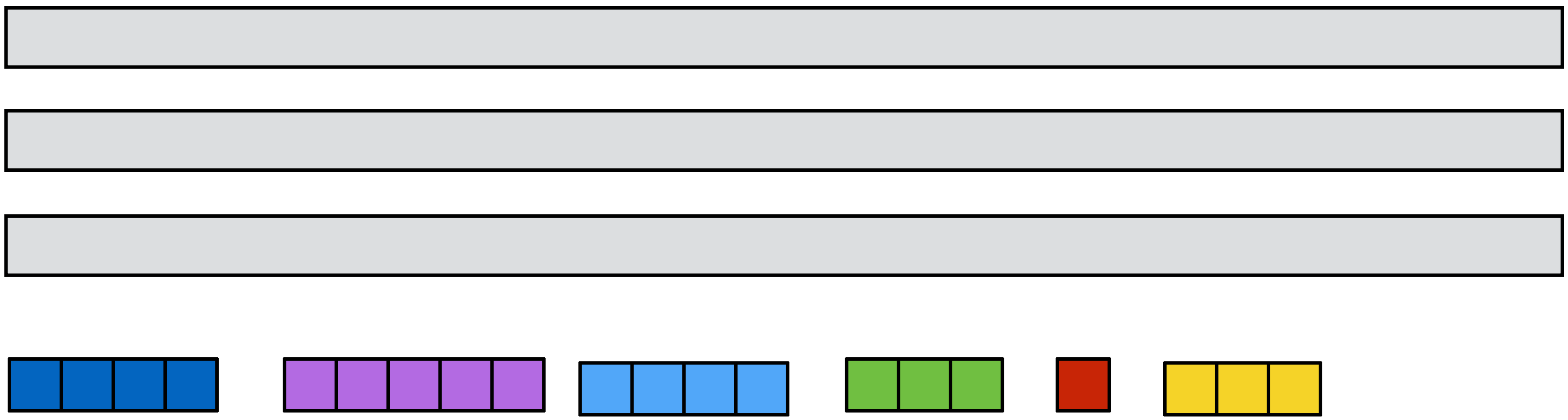


- i: job finishes last.
- All other machines busy until start time s of i. ($s = T_i - t_i$)
- Partition schedule into before and after s.
- After ≤ T*.
- Before:
  - All machines busy => total amount of work = m·s:

$$m \cdot s \leq \sum_i t_i \quad \Rightarrow \quad s \leq \frac{1}{m} \sum_i t_i \leq T^*$$

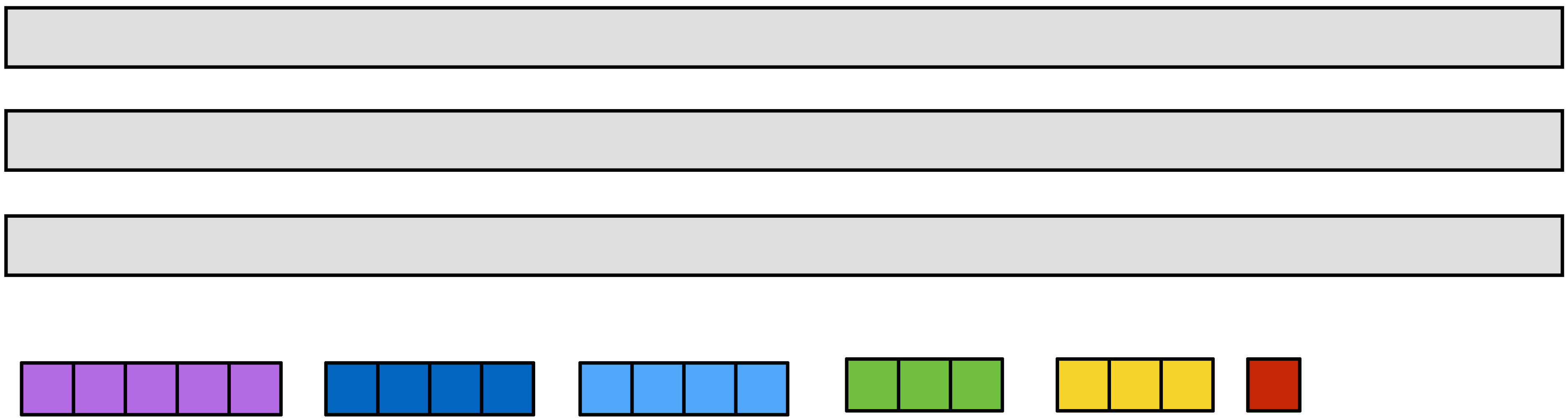
- Length of schedule ≤ 2T*.

Longest processing time rule



- *Longest processing time rule (LPT).* Sort jobs in non-increasing order. Assign next job on list to machine as soon as it becomes idle.

Longest processing time rule



- *Longest processing time rule (LPT).* Sort jobs in non-increasing order. Assign next job on list to machine as soon as it becomes idle.
- LPT is a is a 3/2-approximation algorithm:
  - polynomial time ✓
  - valid solution ✓
  - factor 3/2

Longest processing time rule: factor 3/2

- **Longest processing time rule (LPT).** Sort jobs in non-increasing order. Assign next job on list to machine as soon as it becomes idle.
- Assume $t_1 \geq \ldots. \geq t_n$.
- Lower bound: If $n > m$ then $T^* \geq 2t_{m+1}$.
- Factor 3/2:
  - If $m \leq n$ then optimal.
  - Before $\leq T^*$
  - After: i job that finishes last.
    - $t_i \leq t_{m+1} \leq T^*/2$.
  - $T \leq T^* + T^*/2 \leq 3/2\ T^*$.
- Tight?

Longest processing time rule: factor 4/3

- **Longest processing time rule (LPT).** Sort jobs in non-increasing order. Assign next job on list to machine as soon as it becomes idle.
- Assume $t_1 \geq \ldots. \geq t_n$.
- Assume wlog that smallest job finishes last.
- If $p_n \leq T^*/3$ then $T \leq 4/3\ T^*$.
- If $p_n > T^*/3$ then each machine can process at most 2 jobs in OPT.
- **Lemma.** *For any input where the processing time of each job is more than a third of the optimal makespan, LPT computes an optimal schedule.*

- **Theorem.** *LPT is a 4/3-approximation algorithm.*