# Introduction to scheduling algorithms

Thomas Braun

byte physics e. K.

14th April 2026

# Scheduling theory

- Find an optimal allocation of scarce resources to activities over time
- Exact algorithm: Optimum[1] solution
- $\rho$-approximation algorithm: Solution takes $\rho$ times the optimum solution[2]

[1] Mathematical prove

[2] Some problems are $\mathcal{NP}$-hard so this is the best we can get

# Scheduling problem definition

- Set of $\mathcal{J}$ jobs: $1, \ldots, n$
- Machine environment, each machine can execute at most one job at a time
- Each job requires $p_j$ units of time on the machine
- Completion time of a job: $C_j^S$
- A schedule $\mathcal{S}$ for the set $\mathcal{J}$ specifies, for each job $j$, which $p_j$ units of time the machine uses to process job $j$
- Goal of a scheduling algorithm is to find a „good“[3] schedule

[3] user given optimality criteria

# Examples of optimality criteria

- Minimize completion time of the last job ($C_{max}$)
- Minimize average time, equivalent to ($\sum C_j$)

# More definitions

- preemptive scheduling $\rightarrow$ job is interruptable
- nonpreemptive s. $\rightarrow$ job must be processsed uninterrupted
- Side constraints
  - preemption or not
  - release date (job $j$ is only available starting from $t_j$)
  - precedence constraints (job $j'$ before $j$)
  - weight $w_j > 0 \rightarrow$ optimality criteria: $\sum w_j C_j$
  - due date $d_j$ gives rise to lateness $L_j = C_j^S - d_j \rightarrow$ optimality criteria: minimum of $L_{max} = \max(L_j)$

# Common notation of scheduling problems

- $\alpha|\beta|\gamma$
- $\alpha$: machine environment
- $\beta$: side constraints and other characteristics
- $\gamma$: optimality criterion

# Average completion time

- $1||\sum C_j$
- Shortest processing time (SPT) algorithm is an optimum solution: Order the jobs by nondecreasing processing time (breaking ties arbitrarily) and schedule in that order
- Can be generalized to $1||\sum w_j C_j \rightarrow$ schedule jobs in nonincreasing order of $w_j/p_j$

# Maximum lateness

- $1||L_{max}$
- Earliest due date (EDD) algorithm is an optimum solution: Order the jobs by nondecreasing due dates (breaking ties arbitrarily) and schedule in that order

# How does this relate to polling in Tango?

- Each attribute/command being polled is considered to be a job for **each** read/execute operation
- $\alpha$: We have one machine per device server[4]
- $\beta$: Nonpreemptive jobs, no weights but due dates, but are also „allowed“ to skip jobs in order to accomodate impossible requirements. Polling period is an initial guess for $p_j$ but more realistically is $p_j$ the average time it took to execute the job $\Delta t$
- $\gamma$: Minimum total inaccuracy $\sum I_j = \text{abs}\left(C_j^S - d_j\right)$ and a minimum number of skipped jobs while being fair on skipping

[4] Serialisation by-device, different for by-class/by-process/none

# Bibliography

[1] David Karger (Massachusetts Institute of Technology), Cli Stein (Dartmouth College) und Joel Wein (Polytechnic University). *Scheduling Algorithms*. 2010. URL: https://people.csail.mit.edu/karger/Papers/scheduling.pdf.