Scheduling with restrictions on job positions

Florian Jaehn

Helmut Schmidt University, Hamburg, Germany

Scheduling Seminar, June 24, 2026

Contributors

Antonia Arter
(Wuppertal)

Dirk Briskorn
(Wuppertal)

Maciej Drozdowski
(Poznan)

Julius Hardt
(Hamburg)

Andreas Hipp
(Hamburg)

Minming Li
(Hong Kong)

Radosław
Paszkowski
(Poznan)

Kai Wang
(Hong Kong)

Agenda

- Problems $1|\beta, pd|\gamma$
- Problems $1|\beta, fp|\gamma$

pd (position dependent maintenance): After at most k jobs, the machine must be idle for at least s time units.

fp (fixed position): (must not be combined with preemption) For some jobs, the position within the job sequence is specified.

Definition $1|\beta, pd|\gamma$

- Single machine
- n jobs with processing time p_j and due-dates d_j (if applicable)
- β can be empty or contain (amongst others)
  - ready times r_j
  - preemption (pmtn)
- pd enforces maintenance operations
  - executed after changing *at most* k jobs (or partial jobs) on the machine
  - last for s units of time
  - are non-preemptive
- γ (objective function) e.g.
  - $\sum C_j$
  - $C_{\max} := \max C_j$
  - $L_{\max} := \max\{C_j - d_j\}$

(Fake) Applications

Light Bulb

Aircraft Engine & Landing Gear

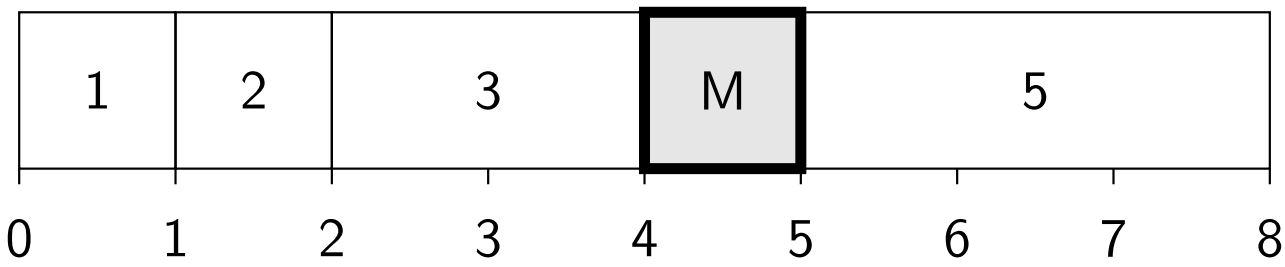
Automotive Battery

Image Credits:
Young asparagus, *An incandescent light bulb*, Wikimedia Commons, licensed under CC BY-SA 4.0.
Xuan Shi Sheng, *ROCAF Fokker 50 5003 Right Wing Main Engine with Landing Gears 20110813*, Wikimedia Commons, licensed under CC BY-SA 3.0.
Shaddack, *Car battery*, Wikimedia Commons, Public Domain (PD).

Example $1|pd|\sum C_j$

$k = 3, s = 1$

j	1	2	3	4
p_j	1	1	2	3

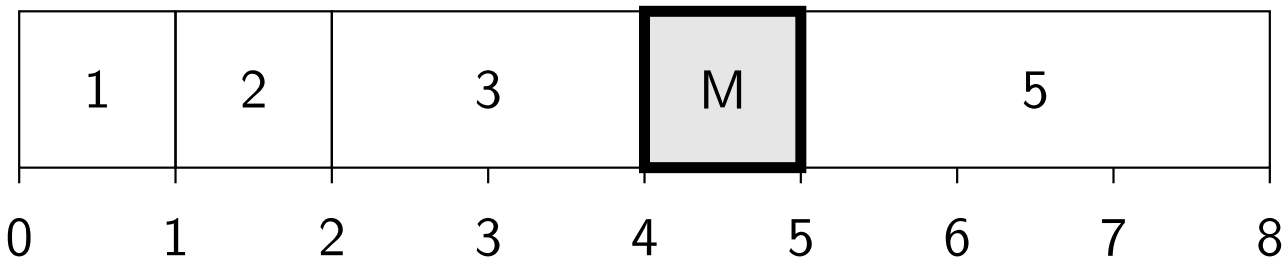


Objective function: $1 + 2 + 4 + 8 = 15$

Example $1|pd|C_{\max}$

$k = 3, s = 1$

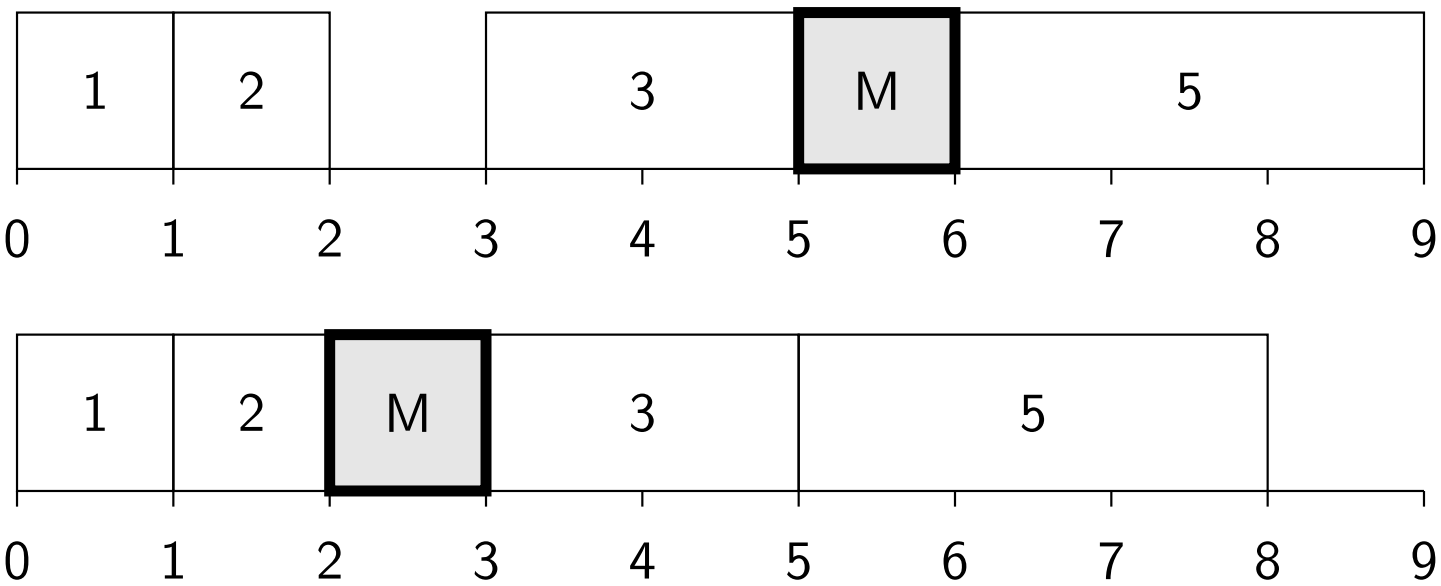
j	1	2	3	4
p_j	1	1	2	3



Example $1|r_j, pd|C_{\max}$

$k = 3, s = 1$

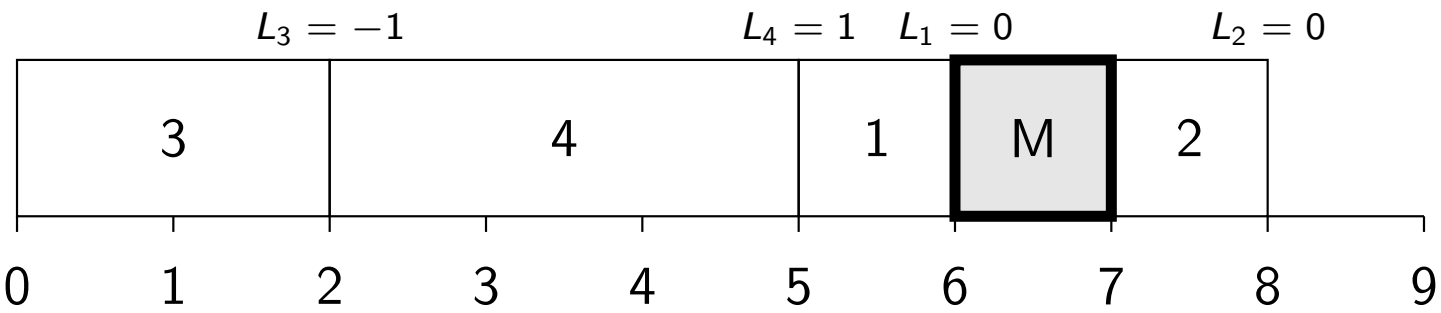
j	1	2	3	4
p_j	1	1	2	3
r_j	0	0	3	3



Example $1|pd|L_{\max}$

$k = 3, s = 1$

j	1	2	3	4
p_j	1	1	2	3
d_j	6	8	3	4



Objective function: $\max\{0, 0, -1, 1\} = 1$

Summary of the Examples

Property

1. $1|pd|\sum C_j$ *is solvable in* $O(n \log n)$
   (SPT-rule, maintenance operations as late as possible)
2. $1|pd|C_{\max}$ *is solvable in* $O(n)$
   (any order of jobs, maintenance operations as late as possible)
3. $1|r_j, pd|C_{\max}$ *is solvable in* $O(n^2)$
   (any order of jobs, DP for scheduling maintenance operations)
4. $1|pd|L_{\max}$ *is solvable in* $O(n \log n)$
   (EDD-rule, maintenance operations as late as possible)

Complexity results remain, even if preemption is allowed.

Is *pd* really boring?

$1|pmtn, pd|L_{\max}$ is easy,

$1|r_j, pmtn|L_{\max}$ is easy, but:

Property $(1|r_j, pmtn, pd|L_{\max})$

Problem $1|r_j, pmtn, pd|L_{\max}$ *is NP-hard in the strong sense.*

Proof.

Easy Proof: Setting $k = n$ and s very large allows us to "forbid" interruptions. $1|r_j|L_{\max}$ reduces to the problem at hand.

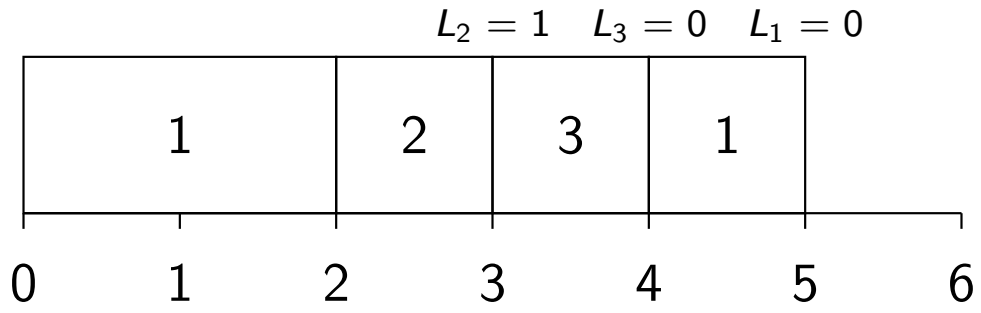
Another proof: shows that the problem is strongly NP-hard for any $k, s \in \mathbb{N}$.

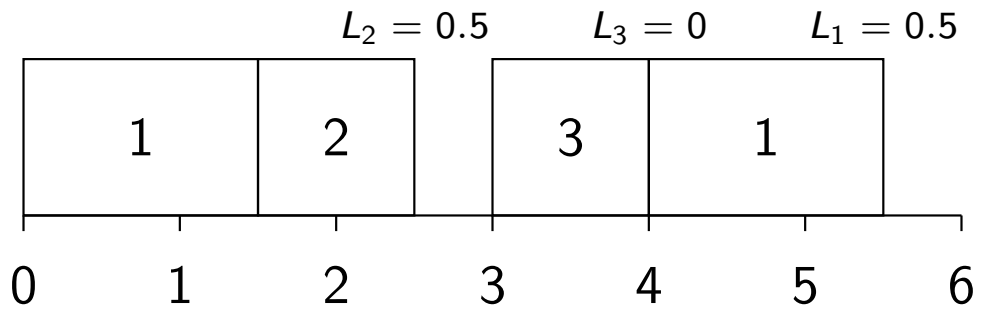
□

Surprising effects for $1|r_j, pmtn, pd|L_{\max}$

$k = 4, s = 1$

j	1	2	3
p_j	3	1	1
r_j	0	1	3
d_j	5	2	4





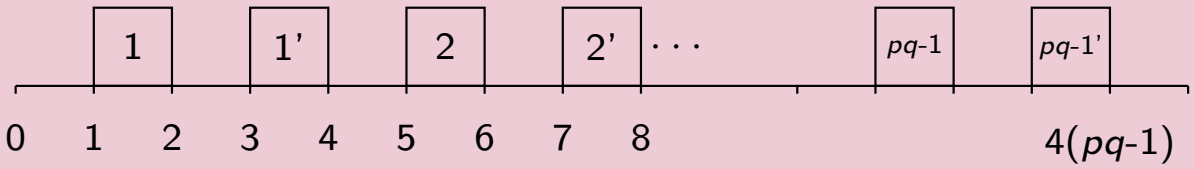
Fractional schedules for $1|r_j, pmtn, pd|L_{\max}$

Property

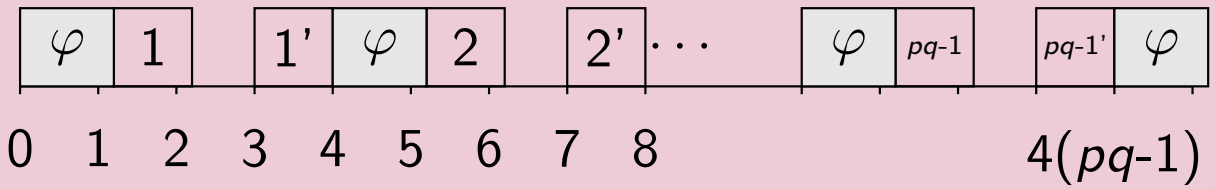
For every rational number $0 < \frac{p}{q} < 1$ *with* $p, q \in \mathbb{N}$*, there is an instance of* $1|r_j, pmtn, pd|L_{\max}$ *with optimal objective function value* $L_{\max} = \frac{p}{q}$.

Proof.

Set k such that there are at most $pq - 1$ interruptions and s large. There are $pq - 1$ pairs (!) of jobs with tight release dates and due dates.



Add another job φ with $r_\varphi = 0$, $d_\varphi = 4(pq - 1) + 1$ and $p_\varphi = pq + p^2$.

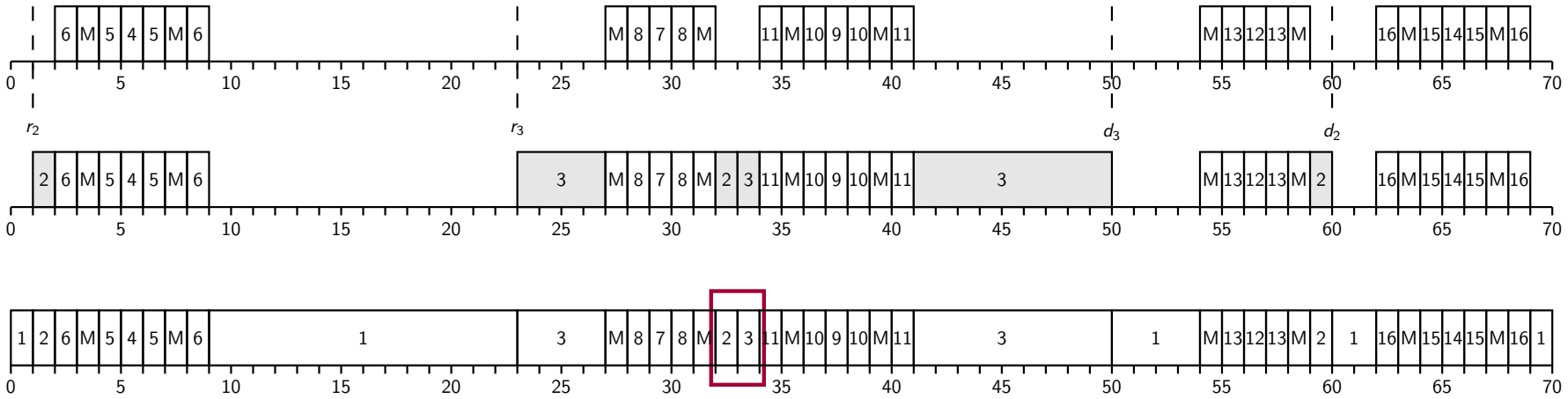


Number of interruptions for $1|r_j, pmtn, pd|L_{\max}$

From for $1|r_j, pmtn|L_{\max}$, we know that there are at most $n-1$ interruptions, each of which is induced by a release date.

Property

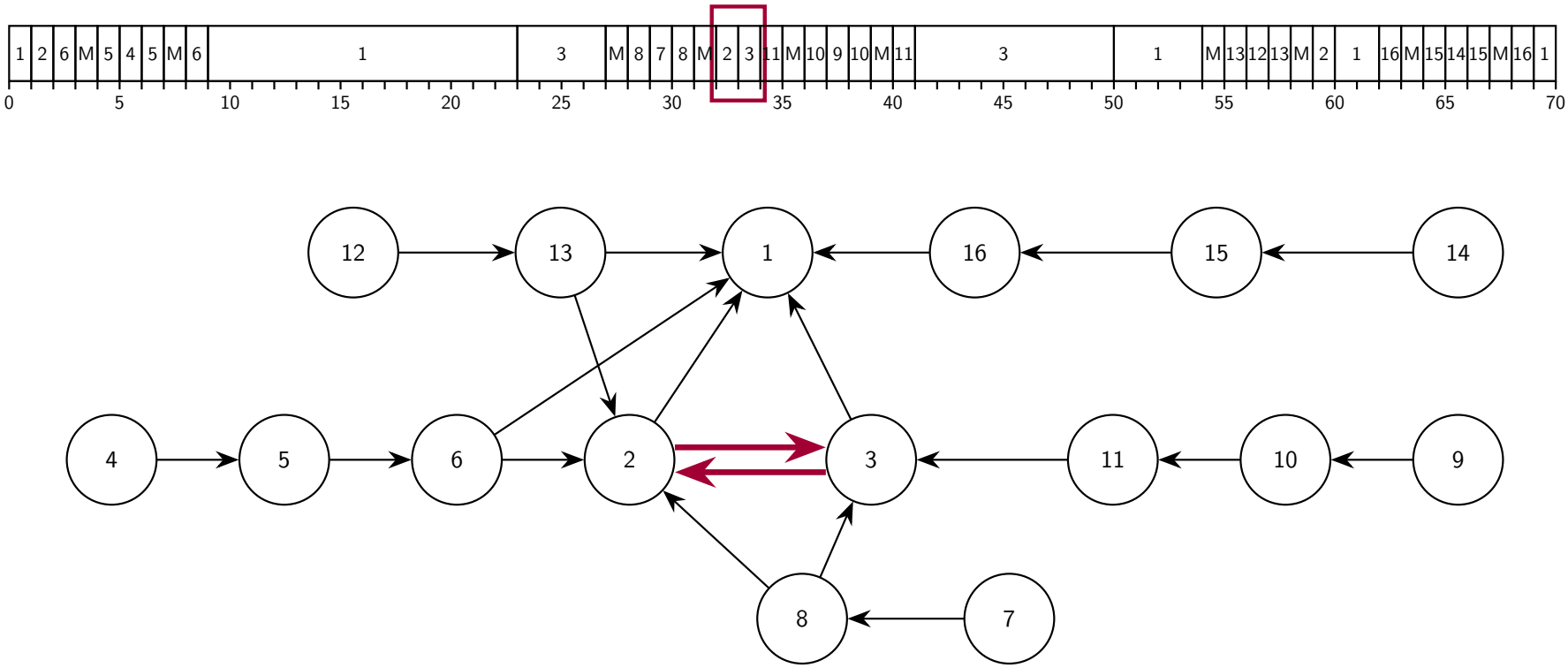
For any $\varepsilon > 0$, there is an instance with more than $\frac{13}{12}n - \varepsilon$ interruptions of jobs in any optimum schedule.



Number of interruptions for $1|r_j, pmtn, pd|L_{\max}$

Property

For any instance $1|r_j, pmtn, pd|L_{\max}$*, there is an optimum solution with at most* $2n-2$ *interruptions of jobs.*



Open questions in position dependent maintenance scheduling

- Tight bounds on the required number of interruptions in preemptive scheduling for various problem settings?
- Multi machine problems are not analyzed so far.

Definition $1|\beta, fp|\gamma$

- Single machine
- n jobs with processing time p_j and due-dates d_j (if applicable)
- fp means:
  - There is a set $Q \subset J$ of special jobs with $q := |Q| < n$.
  - The given injective function, $\eta : Q \to \{1, \ldots, n\}$, fixes the position of a special job in the schedule.
  - A special job j must be scheduled so that exactly $\eta(j)$ regular jobs are completed before j starts.
- γ (objective function) e.g.
  - $\sum C_j$
  - $\sum w_j C_j$
  - $L_{\max} := \max\{C_j - d_j\}$
  - $\sum_{j \in J} U_j$ with $U_j := 1$ if $C_j > d_j$ and $U_j := 0$ if $C_j \leq d_j$

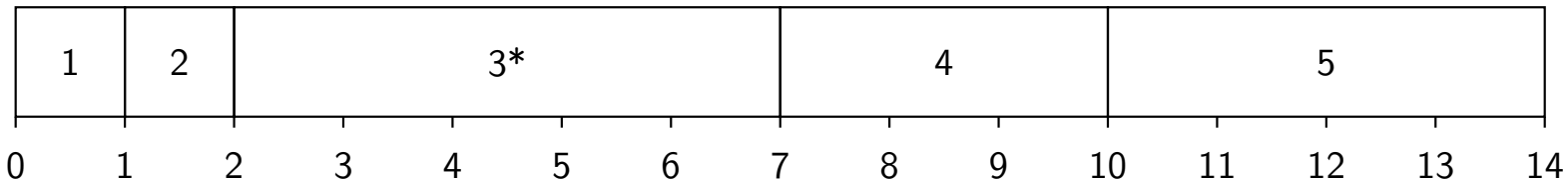
Please note that special jobs also contribute to the value of the objective function.

Applications

- Position dependent maintenance operations that are scheduled as late as possible and that contribute to the objective function value.
- Interim Acceptances in production scheduling.
- Disinfection jobs in Operating Room Scheduling

Example $1|fp|\sum C_j$

j	1	2	3*	4	5
p_j	1	1	5*	3	4



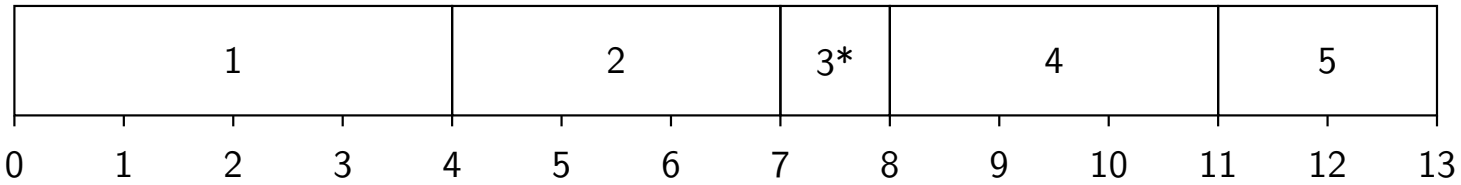
Objective function value: $1 + 2 + 7 + 10 + 14 = 34$

Property

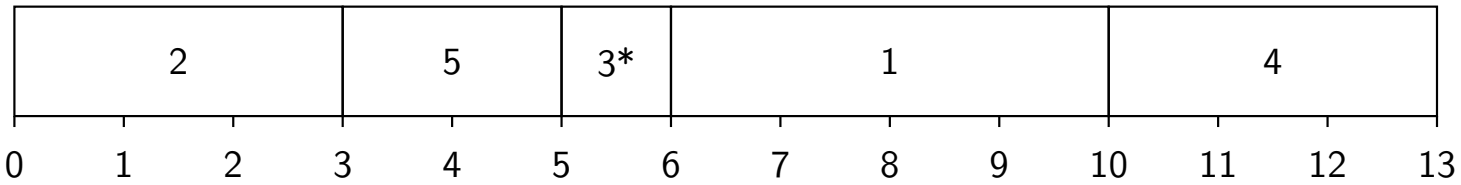
$1|fp|\sum C_j$ *is solvable in* $O(n \log n)$ *(SPT-rule).*

Example $1|fp|\sum w_j C_j$

j	1	2	3*	4	5
p_j	4	3	1*	3	2
w_j	5	3	9*	2	1



Objective function value (WSPT): $4 \cdot 5 + 7 \cdot 3 + 8 \cdot 9 + 11 \cdot 2 + 13 \cdot 1 = 148$



Objective function value (OPT): $10 \cdot 5 + 3 \cdot 3 + 6 \cdot 9 + 13 \cdot 2 + 5 \cdot 1 = 144$

Results for $1|fp|\sum w_j C_j$

Property

Regular jobs, which are not interrupted by special jobs, follow WSPT rule.

Whether or not $1|fp|\sum w_j C_j$ is NP-hard remains an open question, even when there is only one special job.

Property

There is a FPTAS, a $(1+\frac{\varepsilon}{n})$*-approx. with* $O\left(n^{3q+3}(\log W \log P)^{q+1}\left(\frac{1}{\varepsilon}\right)^{2q+2}\right)$ *runtime.*[1]

[1] Unpublished result by Kai Wang

Problem $1||L_{\max}$

Property

Problem $1||L_{\max}$ can optimally be solved using the earliest due date rule (EDD).

Example:

j	1	2	3	4	5	6
p_j	3	5	7	4	2	6
d_j	25	11	8	20	21	15
C_j	3	8	15	19	21	27
L_j	-22	-3	7	-1	0	$\underline{12}$

$\rightarrow$

j	3	2	6	4	5	1
p_j	7	5	6	4	2	3
d_j	8	11	15	20	21	25
C_j	7	12	18	22	24	27
L_j	-1	1	$\underline{3}$	2	$\underline{3}$	2

Problem $1|fp|L_{\max}$

Earliest due date rule is no longer optimal:

j	1	2	3*	4	5	6
p_j	3	5	7*	4	2	6
d_j	25	11	8*	20	21	15
C_j	3	8	15	19	21	27
L_j	-22	-3	7	-1	0	$\underline{12}$

$\xrightarrow{\text{EDD}}$

j	2	6	3*	4	5	1
p_j	5	6	7*	4	2	3
d_j	11	15	8*	20	21	25
C_j	5	11	18	22	24	27
L_j	-6	-4	$\underline{10}$	2	3	2

Optimal solution:

j	2	5	3*	6	4	1
p_j	5	2	7*	6	4	3
d_j	11	21	8*	15	20	25
C_j	5	7	14	20	24	27
L_j	-6	-14	$\underline{6}$	5	4	2

Problem $1|fp|L_{\max}$, Solution

Theorem

The DECISION VERSION of $1|fp|L_{\max}$ can be solved in $O(n^2)$.

So for a given threshold $L_{\max}$, we can decide whether there is a feasible solution. In fact, we can even assume (w.l.o.g.) that $L_{\max} = 0$ by changing due dates accordingly. If we, for example, ask for $L_{\max} = 6$:

j	1	2	3*	4	5	6
p_j	3	5	7*	4	2	6
d_j	25	11	8*	20	21	15

$\rightarrow$

j	1	2	3*	4	5	6
p_j	3	5	7*	4	2	6
d_j	31	17	14*	26	27	21

Problem $1|fp|L_{\max}$, Algorithm

1. Schedule jobs from the end (i.e., from time $P = \sum_{j \in J} p_j$) as follows.
2. If the next job is a special job, schedule this one and set $P = P - p_j$. If it is late, return NO.
3. Else, consider all jobs with $d_j \geq P$. If there are none, return NO. Schedule the longest job j of these jobs and set $P = P - p_j$.
4. Repeat steps 2 and 3 until $P = 0$. If so, return YES.

Example for $L_{\max} = 6$ ($P = 27$):

j	1	2	3*	4	5	6
p_j	3	5	7*	4	2	6
d_j	31	17	14*	26	27	21

$\overset{P=24}{\rightarrow}$

j	2	4	3*	5	6	1
p_j	5	4	7*	2	6	3
d_j	17	26	14*	27	21	31

$P = 20$:

j	2	5	3*	6	4	1
p_j	5	2	7*	6	4	3
d_j	17	27	14*	21	26	31

$\overset{P=14}{\rightarrow}$

j	2	5	3*	6	4	1
p_j	5	2	7*	6	4	3
d_j	17	27	14*	21	26	31

and so on

Problem $1|fp|L_{\max}$, Solution

Theorem

$1|fp|L_{\max}$ *can be solved in* $O(n^2 \log n)$.

Property

Let $P := \sum p_j$ *and* $\underline{L}_{\max} \leq L^*_{\max}$ *be a lower bound. If* $P - d_j < \underline{L}_{\max}$ *holds for a regular job* j*, then job* j *does not induce the objective function value in any (non-idle) solution.*

Conclusion: If the decision problem returns NO for threshold $P - d_j$, i.e., $P - d_j \leq \underline{L}_{\max}$, j can be scheduled in any position. Collect these jobs in set J'.

Property

If $P - d_j > \overline{L}_{\max}$ *holds for a regular job* j *and an upper bound* $\overline{L}_{\max}$*, then job* j *must not be scheduled last in an optimal solution*

Conclusion: If the decision problem returns YES for threshold $P - d_j$, job j must not be scheduled last, but a job from set J'.

Problem $1|fp|L_{\max}$, Algorithm

1. Use lower and upper bounds. Sort regular jobs in non-increasing order of d_j.
2. (Special jobs are scheduled on their intended position.)
3. For $j = 1, 2, \ldots$: check feasibility for $P - d_j$ ($P = \sum p_j$) until job k returns YES. Update bounds and assign $1, \ldots, k-1$ to J'.
4. Schedule the longest job from J' last and call the algorithm with $n - 1$ jobs.

$\underline{L}_{\max} = 3$ (optimal $1||L_{\max}$-solution) $\quad \overline{L}_{\max} = 10$ (EDD-solution) $\quad P = 27$

j	1	5	3*	4	6	2
p_j	3	2	7*	4	6	5
d_j	25	21	8*	20	15	11

$P - d_1 = 2 \leq \underline{L}_{\max} \rightarrow J' = \{1\}$

$P - d_5 = 6 \rightarrow$ YES $\quad$ Job 1 is scheduled last.

Problem $1|fp|L_{\max}$, Algorithm

1. Use lower and upper bounds. Sort regular jobs in non-increasing order of d_j.
2. (Special jobs are scheduled on their intended position.)
3. For $j = 1, 2, \ldots$: check feasibility for $P - d_j$ ($P = \sum p_j$) until job k returns YES. Update bounds and assign $1, \ldots, k-1$ to J'.
4. Schedule the longest job from J' last and call the algorithm with $n - 1$ jobs.

$\underline{L}_{\max} = 3$ $\quad\quad$ $\overline{L}_{\max} = 6$ (Update) $\quad\quad$ $P = 24$

j	5	4	3*	6	2	1
p_j	2	4	7*	6	5	3
d_j	21	20	8*	15	11	25

$P - d_5 = 3 \leq \underline{L}_{\max} \rightarrow J' = \{5\}$

$P - d_4 = 4$ delivers NO $\rightarrow J' = \{4, 5\}$

$P - d_6 = 9 \rightarrow$ YES $\quad\quad$ Job 4 is scheduled last.

Problem $1|fp|L_{\max}$, Algorithm

1. Use lower and upper bounds. Sort regular jobs in non-increasing order of d_j.
2. (Special jobs are scheduled on their intended position.)
3. For $j = 1, 2, \ldots$: check feasibility for $P - d_j$ ($P = \sum p_j$) until job k returns YES. Update bounds and assign $1, \ldots, k-1$ to J'.
4. Schedule the longest job from J' last and call the algorithm with $n - 1$ jobs.

$\underline{L}_{\max} = 4$ $\quad$ $\overline{L}_{\max} = 6$ $\quad$ $P = 20$

j	5	6	3*	2	4	1
p_j	2	6	7*	5	4	3
d_j	21	15	8*	11	20	25

Keep $J' = \{5\}$

$P - d_6 = 5$ delivers NO. Update of the bounds shows: optimal solution found

Summary of the Results

Position-dependent maintenance (pd)

Problems in P: $1|pd|\sum C_j$ $1|r_j, pd|C_{\max}$ $1|pd|L_{\max}$

$1|pmtn, pd|\sum C_j$ $1|pmtn, r_j, pd|C_{\max}$ $1|pmtn, pd|L_{\max}$

NP-hard: $1|pmtn, r_j, pd|L_{\max}$

Fixed-position constrained (fp)

Problems in P: $1|fp|\sum C_j$ $1|fp|L_{\max}$

Open problems: $1|fp|\sum w_j C_j$ $1|fp|\sum U_j$

References

- Arter, A., Briskorn, D., Jaehn, F., et al. (2026). *State-Dependent Machine Availability in Scheduling*. **International Transactions in Operational Research** 33: 3769-3779.
- Drozdowski, M., Jaehn, F., & Paszkowski, R. (2017). *Scheduling Position-Dependent Maintenance Operations*. **Operations Research** 65(6): 25-31.
- Hardt, J., & Jaehn, F. *Bounds and Fractional Processing Times in Position-Dependent Maintenance Operations*. Working Paper.
- Hipp, A., & Jaehn, F. *Single Machine Scheduling Problems with Fixed Job Positions and Uniform Job Characteristics*. Working Paper.
- Jaehn, F. (2024). *Scheduling with Jobs at Fixed Positions*. **European Journal of Operational Research** 318(2): 388-397.

APPENDIX: Problem $1||\sum U_j$

Property

Problem $1||\sum U_j$ can optimally be solved using Moore's algorithm.

1. Initialization: Given an empty set of rejected jobs R.
2. Stop: Schedule all non-rejected (regular) jobs according to earliest due dates. If all non-rejected (regular) jobs are on-time, add all jobs from R to the end of the schedule and stop.
3. Reject job: Let j be the earliest tardy (regular) job, and let i be a (regular) job with longest processing time among the first j jobs of the schedule. Add i to the set of rejected jobs R and go to Step 2.

j	3	2	6	4	5	1
p_j	7	5	6	4	2	3
d_j	8	11	15	20	21	25
C_j	7	12	18	22	24	27
L_j	-1	1	3	2	3	2

$\rightarrow$

j	2	6	4	5	1	3
p_j	5	6	4	2	3	7
d_j	11	15	20	21	25	8
C_j	5	11	15	17	20	27
L_j	-6	-4	-5	-4	-5	19

Problem $1|fp|\sum U_j$, Example

Moore's algorithm is no longer optimal:

j	1	2	3*	4	5	6
p_j	3	5	7*	4	2	6
d_j	25	11	8*	20	21	15
C_j	3	8	15	19	21	27
L_j	-22	-3	$\underline{7}$	-1	0	12

W.l.o.g. we assume that each schedule starts and ends with dummy special jobs of zero length and maximal due date.

Problem $1|fp|\sum U_j$, Property 1

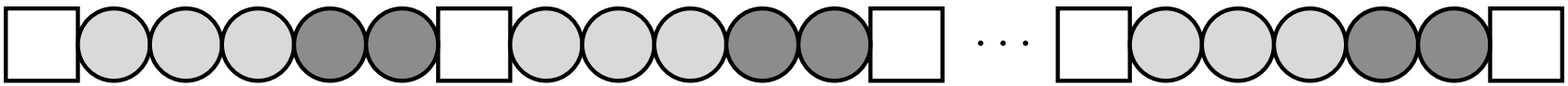
Property

There is always an optimal solution so that between any two special jobs all on-time jobs precede the tardy jobs.

Proof.

Exchange jobs in the "wrong" order. □

Dark grey: tardy jobs
Light grey: on-time job
no filling: tardiness unknown



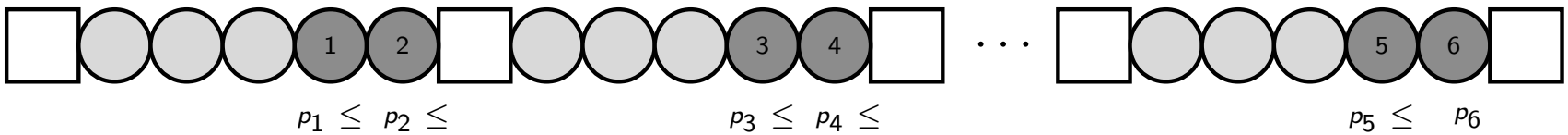
Problem $1|fp|\sum U_j$, Property 2

Property

There is always an optimal solution so that tardy regular jobs are sorted non-decreasingly with respect to processing times.

Proof.

Exchange jobs in the "wrong" order. □



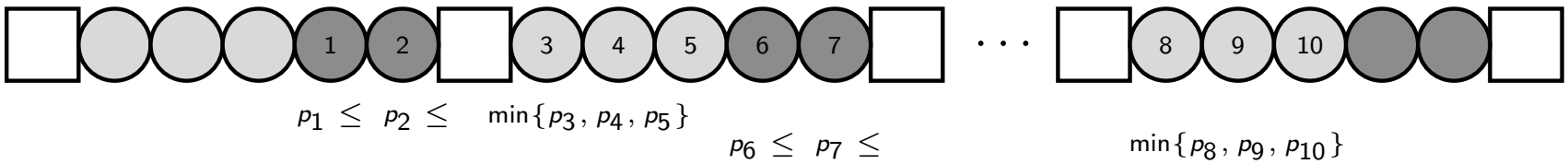
Problem $1|fp|\sum U_j$, Property 3

Property

There is always an optimal solution so that a tardy regular has no longer processing time than all regular on-time jobs that follow in the schedule.

Proof.

Exchange jobs in the "wrong" order. □



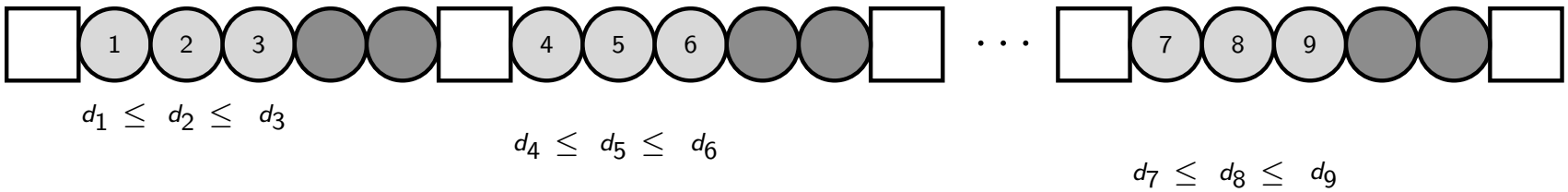
Problem $1|fp|\sum U_j$, Property 4

Property

There is always an optimal solution so that regular on-time jobs, which are not separated by a special job, are sorted according to EDD.

Proof.

Exchange jobs in the "wrong" order. □



Problem $1|fp|\sum U_j$, Definition

Definition

For a regular job $i \in J \backslash Q$ let

$$S[i] := \{j \in J \backslash Q | d_i \leq d_j, p_i > p_j\}$$

be the set of jobs that have no smaller due date but a smaller processing time and let

$$L[i] := \{j \in J \backslash Q | d_i \leq d_j, p_i \leq p_j\}.$$

Note that in Moore's algorithm, a job i is only on-time if all jobs from $S[i]$ are on-time.
Moreover, if the job with smallest due date i and all jobs from $S[i]$ can be scheduled on time, then Moore's algorithm schedules job i first and on-time.

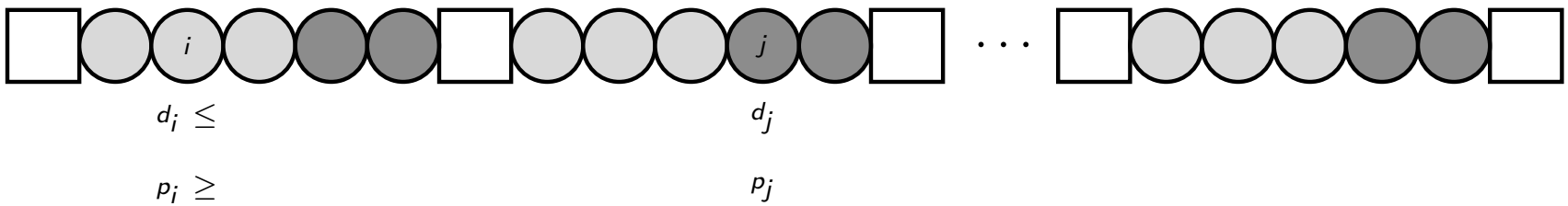
Problem $1|fp|\sum U_j$, Property 5

Property

There is always an optimal solution so that every regular job $i \in J \setminus Q$ is on-time, only if each job $j \in S[i]$ is on-time.

Proof.

Assume for a contradiction that there are only optimal solutions in which job i is on-time but some job $j \in S[i]$ is tardy. As $d_i \leq d_j$, this can only happen if $i \preceq_\sigma j$. However, then we exchange the position of both jobs so that job j is on-time. As $p_j < p_i$, jobs in between the two only advance in time so that the overall objective function does not worsen. This contradicts our assumption. □



Problem $1|fp|\sum U_j$, Example Property 5

If i and $S[i]$ can all be scheduled on-time in problem $1|fp|\sum U_j$, is i then on-time in some optimal solution? NO!

j	1	2	3*	4	5	6	7
p_j	4	2	1*	3	3	5	5
d_j	4	19	7*	10	13	12	17
C_j	4	6	7	10	13	18	23

- Job 1 and $S[1] = \{2, 4, 5\}$ can all be scheduled on-time.
- If Job 1 is on time, there are at least two tardy jobs (note that the last job is always tardy).
- If Job 1 is scheduled last, we obtain a solution with just one tardy job.

j	4	5	3*	6	7	2	1
p_j	3	3	1*	5	5	2	4
d_j	10	13	7*	12	17	19	4
C_j	3	6	7	12	17	19	23

Problem $1|fp|\sum U_j$, What now?

So how can we solve $1|fp|\sum U_j$? I have no idea! :,-(

Let us consider a special case with just one special job $1|fp, q = 1|\sum U_j$.

Property

If there is just one special job, $1|fp|\sum U_j$ can be solved in $O(n^5)$.