Minimizing the weighted number of tardy jobs on a single machine: Strongly correlated instances

Přemysl Šůcha, Lukáš Hejl, Antonín Novák, Zdeněk Hanzálek Czech Technical University in Prague
Czech Institute of Informatics, Robotics and Cybernetics

$$\min\left((t_n - t_1) + \frac{\sum_{\forall(i,j)\in U'} c_{i,j}\left(\bar{p}_{i,j} - p_{i,j}\right)}{1 + \sum_{\forall(i,j)\in U'} c_{i,j}\left(\bar{p}_{i,j} - \underline{p}_{i,j}\right)}\right)$$

s.t.

$$t_j - t_i - p_{i,j} \geq 0$$

$$\sum_{w\in\mathcal{A}} w_w = 1$$

$$W(\omega^{\mathrm{suc}(w)}) < w_w \pi$$

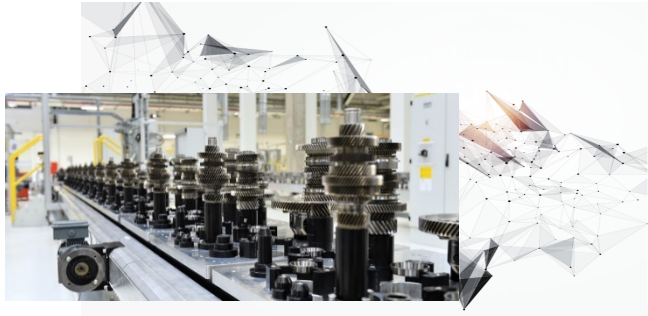
where

$$t_i \in \mathbb{R}_{\geq 0}$$

$$p_{i,j} \in \left[\underline{p}_{i,j}, \bar{p}_{i,j}\right] \cap \mathbb{Z}$$

$$w_w \in [0,1]$$

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE



Czech Technical University in Prague/CIIRC – Optimization group

- Domain:
  - optimization of processes
  - scheduling related to production, health care, embedded systems, human resources, …
  - robust optimization, energy consumption optimization
  - data-driven algorithms, ML for combinatorial problems

- Group is financed from projects (FP6/FP7, H2020, US Navy, …) and cooperation with industry
- Partners:

PORSCHE

EATON

BECKMAN COULTER

ŠKODA

PPL PROFESSIONAL PARCEL LOGISTIC

BLUMENBECKER

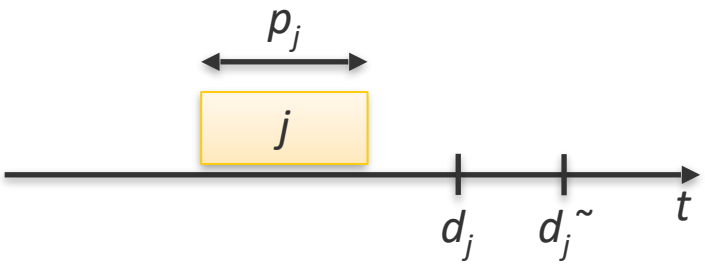
era

Volkswagen

Air Navigation Services of the Czech Republic

$1|d_j^{\sim}|\sum Uj$

- the problem is given by a set of n **jobs** $N=\{1,\ldots,n\}$
- each job $j \in N$ is defined using four non-negative integer parameters:
  - processing time p_j,
  - weight w_j,
  - due date d_j, and
  - deadline $d_j^{\sim}$, $d_j \leq d_j^{\sim}$



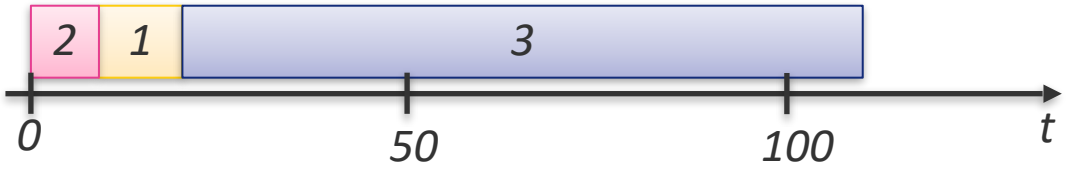
- a solution is a **schedule** - assignment of the jobs to the start times (no overlap, no preemption)
- the goal is to find a schedule minimizing $\sum w_j U_j$ (U_j=1 if the job is tardy, and U_j=0 otherwise).
- in Graham's scheduling notation, the problem is denoted as $1|d_j^{\sim}|\sum w_j U_j$
- The problem is known to be **NP-hard** by Lawler (1983).

$1|d_j\tilde{}|\sum w_jU_j$ - example

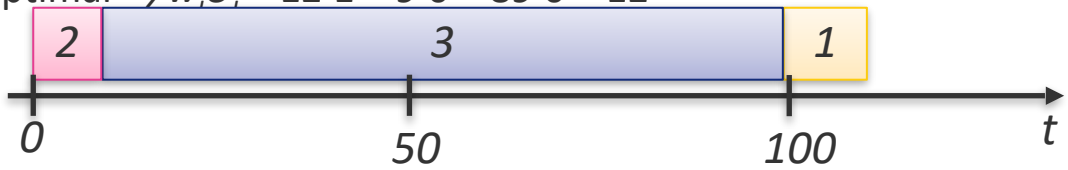
- Example with 3 jobs, i.e., $N=\{1,2,3\}$:

Job j	1	2	3
p_j	11	9	90
w_j	12	9	89
d_j	100	100	100
$d_j\tilde{}$	120	120	120

(a) shortest processing time first - $\sum w_jU_j = 12\cdot0 + 9\cdot0 + 89\cdot1 = 89$



(a) optimal - $\sum w_jU_j = 12\cdot1 + 9\cdot0 + 89\cdot0 = 12$



$1|d_j\sim|\sum w_jU_j$ - literature

- $1||\sum w_jU_j$ is **NP-hard** even if all jobs have a common due date (Karp, 1972)
- $1|d_j\sim|\sum w_jU_j$ **remains NP-hard** even if $\forall j\ w_j = 1$ (Lawler, 1983)
- Sahni (1976) propose dynamic programming-based algorithms with pseudopolynomial time complexity
- Villarreal & Bulfin (1983) published a branch-and-bound algorithms (instances having up to **50** jobs)
- Tang (1990) introduces a new job's dominance (**85** jobs)
- Potts and Van Wassenhove (1988) used an efficient lower bound (**1 000** jobs)
- Knapsack problem was used by M'Hallah and Bulfin (2003) to compute a lower bound (**2 500** jobs)

$1|d_j\tilde{}|\sum w_j U_j$ - literature

- Baptiste et al. (2010) proposed a very efficient branch-and-bound method (**30 000** jobs), nevertheless only 200 jobs in the case of correlated instances ($w_j = p_j + C$)
- Potts and Van Wassenhove (1988) have shown that certain classes of instances of $1||\sum w_j U_j$ and $1|d_j\tilde{}|\sum w_j U_j$ are **significantly harder** to solve
- The same phenomenon was observed with strongly correlated instance of **Knapsack Problem** (Martello & Toth, 1990; Pisinger, 2005)

Outline

- SotA algorithm by Baptiste et al. (2010)
- the hardest instances
- improved algorithm
- experimental results
- conclusions

Algorithm by Baptiste et al. (2010) - properties

- **Dominance Theorem** (Baptiste et al. (2010)). *Let $p_i \leq p_j$, $d_i \geq d_j$, $\tilde{d_i} \leq \tilde{d_j}$ and $w_i \geq w_j$, and at least one inequality is strict. Then*
  - *if job i is tardy, then job j must be tardy too,*
  - *if job j is early, then job i must be early too.*

- **Reduction Theorem:** (Baptiste et al. (2010)). *There exists a feasible schedule with early set E if and only if there exists a feasible schedule with early set $E = E \setminus \{i\}$ for the reduced problem.*

Algorithm by Baptiste et al. (2010) - ILP

- the algorithm exploits an **ILP formulation**
- **binary variable** x_i equals to one if $i \in E$ and zero otherwise
- the objective **maximizes the weighted number of early jobs** ($\approx$ min $\sum w_j U_j$)
- the constraint defines two sets:
  - jobs that must be **completed** before t : $A_t = \{i \in N : d_i^{\sim} \leq t\}$
  - jobs that will be **early** if they are scheduled before t : $B_t = \{i \in N : d_i \leq t \wedge d_i^{\sim} > t\}$

$$\max_{\boldsymbol{x}} \sum_{i \in N} w_i x_i,$$

subject to

$$\sum_{i \in A_t} p_i + \sum_{i \in B_t} p_i x_i \leq t, \quad t \in T,$$

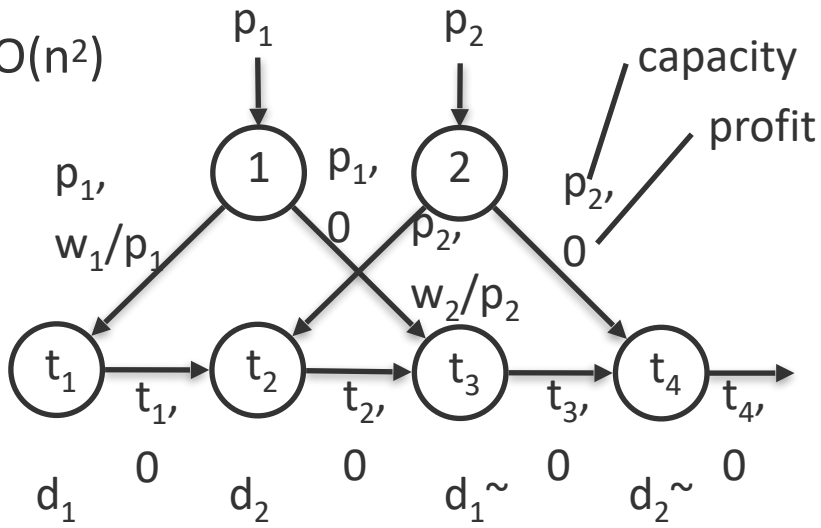
$$x_i \in \{0, 1\}, \quad i \in N.$$

Algorithm by Baptiste et al. (2010)

- The algorithm is a branch and bound, deciding if a job is early or tardy (reduces $\mathbf{1|d_j^{\sim}|\sum w_jU_j}$ to $\mathbf{1|d_j^{\sim}=D_j|-}$ which is polynomial)
- Each node of the branch-and-bound tree is processed as follows:
  1. (**Upper bound** z^{-}): the algorithm solves the problem relaxation,
  2. (**Lower bound** $\underline{z}$): the heuristic computing to a feasible solution
  3. (**Fixing of decisions**): variable fixing techniques deciding is a job is early or tardy,
  4. (**Branching**): the algorithm selects job i and recursively branches with $i \in E$ and $i \in N \setminus E$ (finished by ILP)

Algorithm by Baptiste et al. (2010) – upper bound

- LP relaxation of the ILP model is **too large to fit into memory**
- the algorithm transforms the LP relaxation into **maximum profit flow** problem (solved by dual ascent method)
- There are two types of nodes: **jobs** and **time**
- memory complexity is O(n) compared to $O(n^2)$
- the objective is used for **solution pruning**



Algorithm by Baptiste et al. (2010) - lower bound

- the lower bound is computed by a **heuristic**
- the algorithm:
    1. use the solution obtained by solving the **relaxed problem** (*y*),
    2. based on *y* fix decisions of some jobs using the **reduction theorem**,
    3. the remining jobs are **decided by the ILP**,
    4. the resulting solution is improved by a **local search** (difference is 2)

Algorithm by Baptiste et al. (2010)

- variable fixing
  - performs job fixing ($i \in E$ or $i \in N \setminus E$)
  - uses **variable-fixing techniques** from Integer Linear Programming
  - Method exploits the **lower and upper bounds** and **reduced cost** of variables
- Branching
  - **depth-first search** branch and bound
  - branching on the fractional variable having largest max–min pseudo-cost
  - if the reduced problem allows to build an instance of the ILP with no more than $1.4 \cdot 10^7$ nonzeros, the **rest is solved by the ILP solver**

$1|d_j^{\sim}|\sum w_j U_j$ – correlated instances

- Potts and Van Wassenhove (1988) defined three classes of instances regarding the relationship between w_j and p_j as:
  - ***strongly correlated*** ($w_j = p_j + C$, where C is often 20),
  - ***weakly correlated*** (w_j is drawn from the uniform distribution $w_j \sim [p_j, p_j + C]$,
  - ***uncorrelated*** (do not have any specific relation between p_j and w_j)
- Algorithm by Baptiste et al. (2010) cannot solve some **instances with 250 jobs** of strongly correlated instances (all for **30 000 jobs** of uncorrelated ones)
- Note that for strongly correlated instances the condition in the **dominance theorem** reduces to *"*$p_i = p_j$, $d_i \geq d_j$, $d_i^{\sim} \leq d_j^{\sim}$ *and* $w_i = w_j$*"* – harder to fulfill

Improved algorithm - Motivation

- the limiting factor for solving uncorrelated and correlated instances in the algorithm Baptiste et al. (2010) is not the same:
  - uncorrelated instances
    - the limiting factor is the **memory limit** (quadratic size of ILP)
  - correlated instances
    - the **CPU time** of the algorithm grows much faster (job's dominance property)
    - **variable-fixing technique** is less efficient (four times more jobs without decision)
- we propose **three improvements** of the algorithm

Improved algorithm - ILP

- The improved ILP reformulates the objective function for correlated instances and introduces variable $e = |E|$
- ILP is **decomposed to subproblems** according to e
- a sub-problem can be seen as an instance where $w_i = p_i$ (the condition in Theorem 1 reduces to $p_i = p_j$, $d_i \geq d_j$ and $d_i^{\sim} \leq d_j^{\sim}$
- moreover, it is not needed to assume all e
- the **lower bound of *e*** is obtained by solving

  $1|d_j^{\sim}, w_j' = p_j|\sum w_j U_j$
- the **upper bound of *e*** is obtained by solving

  $1|d_j^{\sim}|\sum U_j$

$$\max_{e \in \{0,\ldots,n\}} \max_{\boldsymbol{x}} \left(C \cdot e + \sum_{i \in N} p_i x_i \right),$$

subject to

$$\sum_{i \in A_t} p_i + \sum_{i \in B_t} p_i x_i \leq t, \quad t \in T,$$

$$\sum_{i \in N} x_i = e,$$

$$x_i \in \{0, 1\}, \quad i \in N.$$

Improved algorithm - Lower bound

- Both bounds are used in the variable fixing technique – the tighter bounds the smaller N
- we proposed main three improvements:
  - use of **decomposed ILP model**
  - **reversed condition** for defining which jobs will be solved by ILP
  - we do not use the additional **local search**
- The modified condition increased the number of jobs solved by ILP (**13%**) – compensated by leaving out the additional local search

Improved algorithm - Upper bound

- the upper bound proposed in Baptiste et al. (2010) is already very tight
- we tighten up the obtained upper bound by **branching on a selected job**:
  - branching on a job being early/tardy
  - branching on a partially scheduled job with minimal d_j
    - simple rule
    - significant impact on the objective
  - branching is repeated up to a **small fixed depth** (take the worst one)
- The tighter lower and upper bounds we provide to the variable fixing technique, the fewer nodes it is necessary to explore

Experimental results – original ILP vs. our ILP

Strongly correlated instances with $C = 20$ solved using ILP models.

n	original ILP model Baptiste et al. (2010)			our ILP model		
	CPU time		unsolved	CPU time		unsolved
	avg [s]	max [s]	out of 200 [-]	avg [s]	max [s]	out of 200 [-]
50	0.03	0.31	0	0.10	0.40	0
100	1.42	221.72	0	0.17	1.17	0
150	32.18	3600.00	1	0.28	1.42	0
200	174.81	3600.00	6	0.37	3.00	0
250	444.72	3600.00	22	0.48	1.80	0
500	579.14	3600.00	30	1.82	11.65	0
1000	1153.90	3600.00	63	9.01	64.01	0
2000	1123.90	3600.00	61	46.79	423.79	0
3000	1328.78	3600.00	72	153.44	1173.26	0
4000	1642.63	3600.00	90	285.79	2567.63	0
5000	1586.28	3600.00	86	432.04	3022.05	0

Experimental results

- B&B

Strongly correlated instances with $C = 20$ and deadlines.

	Results of the original algorithm			Results of our algorithm		
	unsolved	CPU time		unsolved	CPU time	
n	out of 200 [-]	avg [s]	max [s]	out of 200 [-]	avg [s]	max [s]
1000	18	330.91	3600.00	0	1.57	16.15
2000	29	558.35	3600.00	0	3.93	46.52
3000	47	867.64	3600.00	0	7.38	48.37
4000	44	849.17	3600.00	0	14.33	332.31
5000	47	875.12	3600.00	0	17.53	94.21
6000	49	918.11	3600.00	10	218.03	3600.00
7000	68	1249.09	3600.00	32	598.27	3600.00
8000	71	1318.14	3600.00	38	733.45	3600.00
9000	97	1775.35	3600.00	64	1189.44	3600.00
10000	103	1870.18	3600.00	68	1255.93	3600.00
200	0	3.56	353.49			
250	6	131.15	3600.00			

Conclusions

- the work studies problem $1|d_j^{\sim}|\sum w_j U_j$ on **the hardest problem instances**
- the main idea is the **decomposed ILP method** + few simple improvements
- The SotA algorithm cannot solve all instances with 250 jobs (within an hour)
- our approach can solve all instances with **5 000 jobs**
- an improvement can be observed on **weakly and uncorrelated instances**
- our original aim was to use ML to improve the algorithm – size is probably too large

Lukáš Hejl, Přemysl Šůcha, Antonín Novák, Zdeněk Hanzálek, Minimizing the weighted number of tardy jobs on a single machine: Strongly correlated instances, *European Journal of Operational Research*, Volume 298, Issue 2, Pages 413-424, 2022.