A tour through a maze of project and machine scheduling problems

Tamás Kis
Senior research fellow at the Institute of Computer Science and Control, Hungary

October 18, 2016

Who I am?

- PhD in Mathematics at Ecole Polytechnique Fédérale de Lausanne, Suisse (2001)
- Oui, je parle un peu français
- Main areas of research
  - Machine scheduling
  - Project scheudling
  - Integer programming

A non-exhaustive list of problems...

- The car-sequencing problem
- *Project scheduling with variable intensity activities
- Project scheduling with earliness/tardiness costs
- *Resource leveling in a machine environment
- Large scale multi-mode job-shop type problems
- *Bilevel scheduling problems
- Integrated planning and scheduling problems
- Scheduling with energy constraints (CECSP) (collaboration with LAAS)
- *Machine scheduling with non-renewable resources
- *Resource-constrained shortest path problem
- Vehicle and crew scheduling problems
- Disjunctive cuts and integer programming

Scheduling with variable intensity activities and feeding precedence constraints

Outline

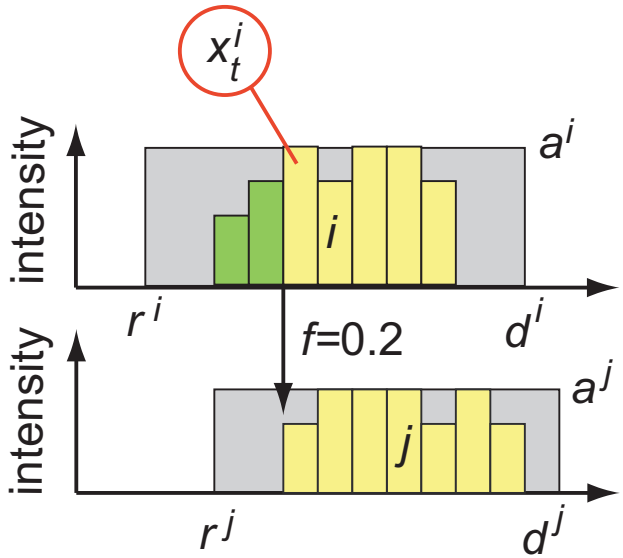
- Problem data and MILP formulation
- Computational complexity
- Polyhedral results
- Branch-and-Cut
- Computational Results
- Applications

Problem Data (1)

- Discrete time horizon: $t = 1, \ldots, T$
- Continuously divisible renewable resources: $k \in R$
  - normal capacity: $b_t^k \geq 0$
  - extra capacity: $\overline{b}_t^k \geq 0$
  - cost of using extra capacity: $c_t^k \geq 0$
- Variable intensity activities: $i \in N$
  - maximum intensity: $0 < a^i \leq 1$
  - time window: $\{r^i, \ldots, d^i\}$
  - resource requirements: $q_k^i \geq 0$, $k \in R$

Problem Data (2)

- Feeding Precedence Constraints: $(i, j, f) \in A$
  - At least f fraction of activity i must be completed before j may start
  - The fraction of activity j completed up to any period τ never exceeds the fraction of activity i completed up to period τ, i.e., $\sum_{t=r^i}^{\tau} x_t^i \geq \sum_{t=r^j}^{\tau} x_t^j, \forall\ \tau$



The MILP formulation (1)

Variables:

$$
\begin{aligned}
x_t^i &= \textit{intensity} \text{ of activity } i \text{ in period } t \\
z_t^{i,f} &= f\textit{-fraction mask} \text{ of activity } i \text{ in period } t,\ z_t^{i,f} \in \{0,1\} \\
y_t^k &= \text{external capacity of resource } k \text{ used in period } t
\end{aligned}
$$

Objective:

$$\min \sum_t \sum_{k \in R} c_t^k y_t^k$$

The MILP formulation (2)

Constraints:

$$\sum_{t=r^i}^{d^i} x_t^i = 1, \quad i \in N \tag{1}$$

$$0 \leq x_t^i \leq a^i, \quad i \in N,\ t \in \{r^i, \ldots, d^i\} \tag{2}$$

$$\sum_{i \in N,\ t \ni \{r^i, \ldots, d^i\}} q_k^i x_t^i \leq b_t^k + y_t^k, \quad k \in R, t \in \{1, \ldots, T\} \tag{3}$$

$$0 \leq y_t^k \leq \overline{b}_t^k, \quad k \in R,\ t \in \{1, \ldots, T\} \tag{4}$$

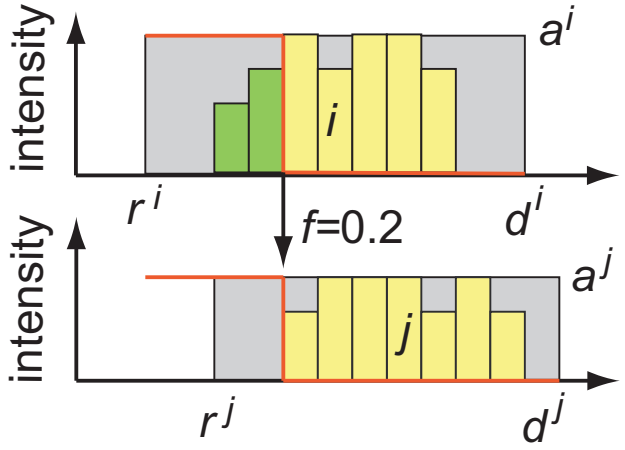
The MILP formulation (3)

$$\sum_{t=r^j}^{\tau} x_t^j - \sum_{t=r^i}^{\tau} x_t^i \leq 0, \quad (i,j,f) \in A,\ \tau \in \{r^i, \ldots, d^i\} \tag{5}$$

$$\sum_{t=r^i}^{\tau-1} x_t^i + f \cdot z_\tau^{i,f} \geq f,\ (i,j,f) \in A,\ \tau \in \{r^i + p^{i,f}, \ldots, d^i\} \tag{6}$$

$$\sum_{t=\tau+1}^{d^j} x_t^j - z_\tau^{i,f} \geq 0, \quad (i,j,f) \in A,\ \forall \tau \tag{7}$$

$$z_t^{i,f} - z_{t+1}^{i,f} \geq 0, \quad \forall i, f, t \tag{8}$$



Other objective functions

Below we assume that there are no extra capacities: $\overline{b}_t^k = 0$

- Min Makespan: by dichotomic search find the smallest T such that a feasible solution exists with $d^i = T$.
- Min Maximum Tardiness: $d^i = \tilde{d}^i + T_{\max}$ and dichotomic search on $T_{\max}$, where $\tilde{d}^i$ is the due-date of $i \in N$
- Min Weighted Total Tardiness: define weights $w_t^i = 0$ if $t \in \{r^i, \ldots, \tilde{d}^i\}$ and $w_t^i = w^i$ if $t \in \{\tilde{d}^i + 1, \ldots, T\}$. The objective is min $\sum_{i \in N} \sum_{t \in \{r^i, \ldots, T\}} w_t^i z_t^i$.

Computational complexity

- If extra capacity is unlimited, a feasible solution can be found in polynomial time.
  - Heuristics often assume and exploit that a feasible solution can be found quickly
- If extra capacity is bounded, then feasible solution existence is NP-complete in the strong sense.

Cutting planes from precedence constraints

- Relaxation: drop all resource constraints (RIP)
- Feasible solutions of RIP = all intensity assignments to activities respecting the precedence constraints
- A valid inequality for RIP is valid for the original IP
- LP-RIP, the LP relaxation of RIP, has vertices with fractional z coordinates
- LP-RIP can be tightened by automatically generating valid inequalities

The polytope K^{ijf} (1)

Consider $(i, j, f) \in A$ and let the polytope K^{ijf} be the convex hull of those $(x^i, x^j, z^{i,f})$ satisfying the following:

$$\sum_{t=r^i}^{d^i} x_t^i = 1, \tag{9}$$

$$0 \leq x_t^i \leq a^i, \quad t \in \{r^i, \ldots, d^i\} \tag{10}$$

$$\sum_{t=r^i}^{\tau-1} x_t^i + f \cdot z_\tau^{i,f} \geq f, \quad \tau \in \{r^i + p^{i,f}, \ldots, d^i\} \tag{11}$$

$$z_t^{i,f} - z_{t+1}^{i,f} \geq 0, \quad t \in \{r^i + p^{i,f}, \ldots, d^i - 1\} \tag{12}$$

$$z_t^{i,f} \in \{0, 1\} \tag{13}$$

The polytope K^{ijf} (2)

$$\sum_{t=r^j}^{d^j} x_t^j = 1, \tag{14}$$

$$0 \leq x_t^j \leq a^j, \quad t \in \{r^j, \ldots, d^j\} \tag{15}$$

$$\sum_{t=\tau+1}^{d^j} x_t^j - z_\tau^{i,f} \geq 0, \quad \tau \in \{r^i + p^{i,f}, \ldots, d^i\} \tag{16}$$

(To simplify notation, assume $r^i + p^{i,f} = r^j$ and $d^i + p^j \leq d^j$.)

Results for K^{ijf}

Decompose K^{ijf} into two smaller dimensional polytopes:

$$\begin{aligned} K_i^{ijf} &= \{(x^i, z^{i,f}) \mid (x^i, z^{i,f}) \text{ satisfies } (9) - (13)\} \\ K_j^{ijf} &= \{(x^j, z^{i,f}) \mid (x^j, z^{i,f}) \text{ satisfies } (12) - (16)\} \end{aligned}$$

Lemma $(x^i, x^j, z^{i,f}) \in K^{ijf}$ if and only if $(x^i, z^{i,f}) \in K_i^{ijf}$ and $(x^j, z^{i,f}) \in K_j^{ijf}$

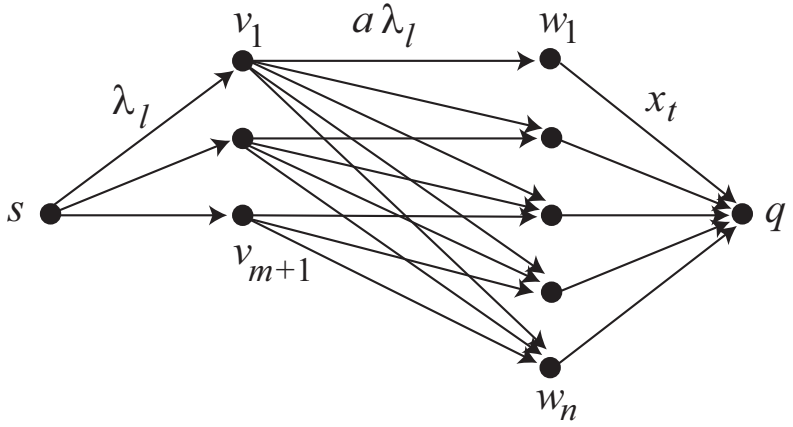
Observation Both K_i^{ijf} and K_j^{ijf} are instances of the polytope $K^{f'}$:

$$K^{f'} = \text{conv}\{(x, z) \in \mathbb{R}^n \times \mathbb{B}^m \mid \sum_{t=1}^{n} x_t = 1,\ \forall t : 0 \leq x_t \leq a$$

$$\forall \tau : \sum_{t=\tau+1}^{m} x_t - f' \cdot z_\tau \geq 0,\ \forall t : z_t \geq z_{t+1}\}$$

Results for K^f (1)

- If $f = 1$, a minimal linear representation of K^f is known. Cuts derived from a network flow



  $\lambda_1 = 1 - z_1$, $\lambda_\ell = z_{\ell-1} - z_\ell$ and $\lambda_{m+1} = z_m$
- The non-trivial inequalities are of the form:

$$a_{\mathrm{rem}} z_{t_1} + \sum_{t \in S_1 \setminus \{t_1\}} a z_t \leq \sum_{t \in \{t_1, \ldots, n\} \setminus (S_1 \cup S_2)} x_t,$$

  where $(p-1)a < 1 \leq pa$, $a_{\mathrm{rem}} = 1 - (p-1)a$, $t_1 = \min S_1$, $S_1 \subseteq \{1, \ldots, m\}$, $S_2 \subset \{m+1, \ldots, n\}$ and $|S_1| + |S_2| = p$
- There is an $O(n \log n)$ algorithm for separating (S_1, S_2) inequalities

Results for K^f (2)

- When $0 < f \leq 1$, then the non-trivial facets of K^f are

$$h_r z_{t_1} + h \sum_{t \in S_1 \setminus \{t_1\}} z_t + \sum_{t \in \{1,\ldots,t_1\} \setminus (S_1 \cup S_2)} x_t \quad \geq \quad f - h|S_2|,$$

for every $\emptyset \neq S_1 \subseteq \{p+1, \ldots, m\}$, $S_2 \subset \{1, \ldots, p\}$, $|S_1| + |S_2| = p$, and t_1 being the greatest element of S_1;

$$(f - 1 + h|U_\ell^2|)z_\ell + h \sum_{t \in U_\ell^1} z_t + \sum_{t \in \{1,\ldots,n\} \setminus (U_\ell^1 \cup U_\ell^2)} x_t \geq f,$$

for $\ell \in \{p+1, \ldots, m\}$, $U_\ell^1 \subseteq \{p+1, \ldots, \ell\}$, $U_\ell^2 \subseteq \{\ell+1, \ldots, n\}$, $h|U_\ell^1 \cup U_\ell^2| \leq 1$, and $h|U_\ell^2| \leq 1 - f$;

$$h \sum_{t \in U} z_t + \sum_{t \in \{1,\ldots,m\} \setminus U} x_t \geq f,$$

for $U \subseteq \{p+1, \ldots, m\}$ with $h|U| \leq 1$.

Test Instances

- De Boer instances:
  - $f = 1$
  - randomly generated precedence constraints
  - $1/a^i$ between 1 and 5
  - $n = 10, 20$ or 30 activities
  - $r = 3, 5$ or 20 resources
  - average $d^i - r^i - p^i + 1$ value: $s = 2, 5, 10, 15$ or 20 (over the activities of the instance)
  - extra capacity is unlimited ($\overline{b}_t^k = \infty$)
  - 10 instances with the same n, r and s parameters
- My computational environment: PC, Pentium 4, 1.6GHz
- Algorithm B: Branch-and-Cut truncated after 675 seconds

Comparison to Branch-and-Price

Algorithm H:
E. Hans, *Resource loading by branch-and-price techniques*,
Ph. D. thesis, Twente University Press, The Netherlands, 2001.

Table The average of $ub(B)/ub(H)$

	$n = 10$			$n = 20$			$n = 50$		
s	$r = 3$	10	20	3	10	20	3	10	20
2	1	1	1	1	1	1	1	1	1
5	1	1	1	0.99	0.99	0.99	0.95	0.96	0.99
10	1	1	0.99	0.86	0.97	0.98	0.73	0.94	0.97
15	0.88	0.98	0.98	0.63	0.94	0.97	0.63	0.92	1.03
20	0.89	0.98	0.98	0.71	0.91	0.97	0.56	0.92	0.96

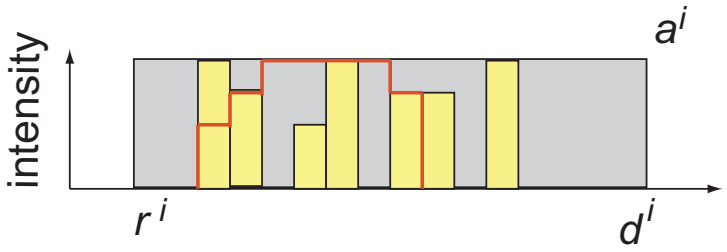
Results on instances with $0 < f < 1$

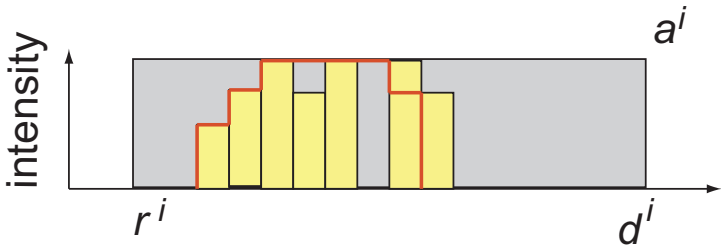
For $f = 0.5$, average ub/lb values for algorithm B^+ (first row), and B^- (second row), respectively, in each class (n, r, s).

	$n = 10$			$n = 20$			$n = 50$		
s	$r = 3$	10	20	3	10	20	3	10	20
2	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1.02	1.06
	1	1	1	1	1	1	1.02	1.03	1.07
20	1	1	1	1	1	1.01	1.01	1.37	1.17
	1	1	1	1	1.01	1.01	1.01	1.5	1.16

Application to Production Planning

- The first optimal solution with cost $C^* = \sum_{t,k} c_t^k y_t^k$ is usually unsatisfactory





- Push the activities to the left (or to the right) by minimising $\sum_{i,t} w_t^i z_t^i$ with appropriate weights w_t^i under the additional constraint $\sum_{t,k} c_t^k y_t^k \leq C^*$
- Transform the solution to this problem by e.g., solving a linear program with quadratic objective function after fixing the $z_t^{i,f}$ to the values found previously.

Publications

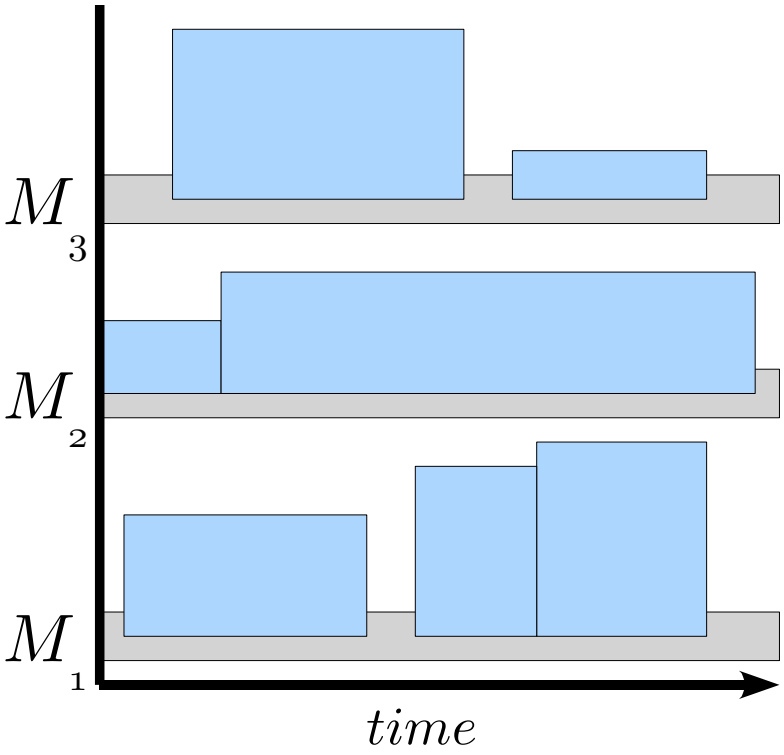
- Kis, T. (2005). A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. Mathematical programming, 103(3), 515-539.
- Kis, T. (2006). Rcps with variable intensity activities and feeding precedence constraints. In Perspectives in modern project scheduling (pp. 105-129). Springer US.

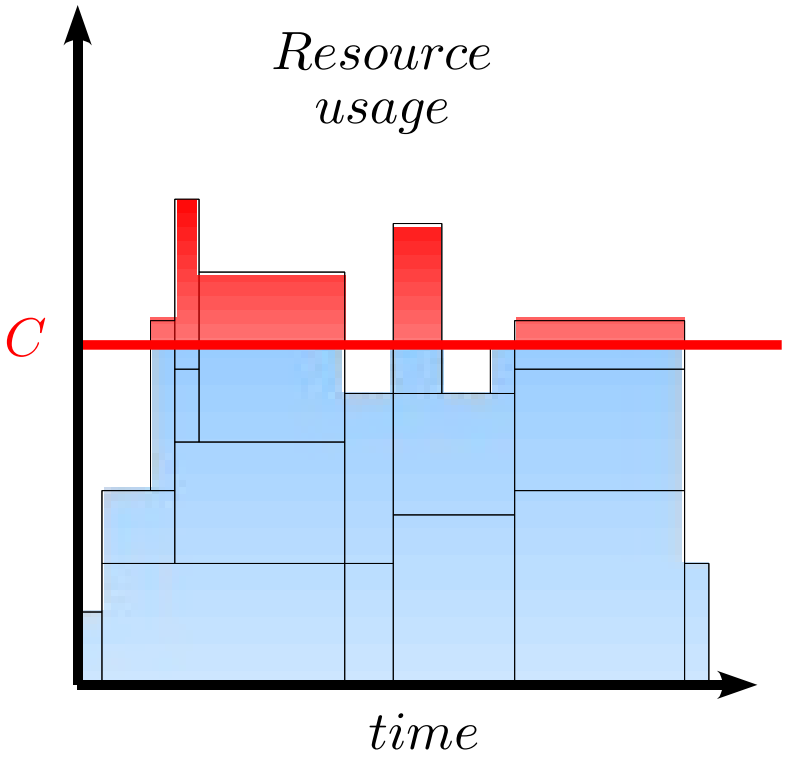
Resource leveling in a machine environment

joint work with Márton Drótos

Outline

Resource usage in a machine scheduling problem



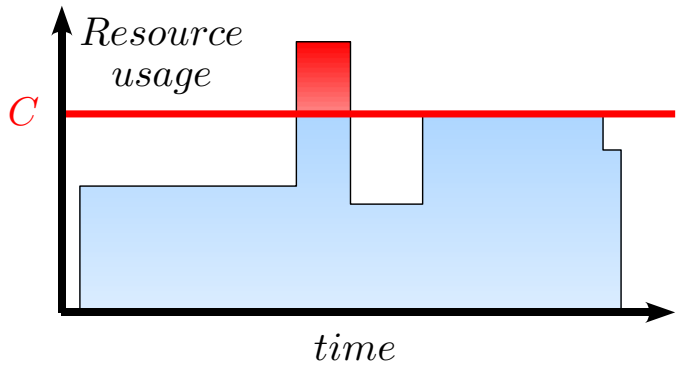


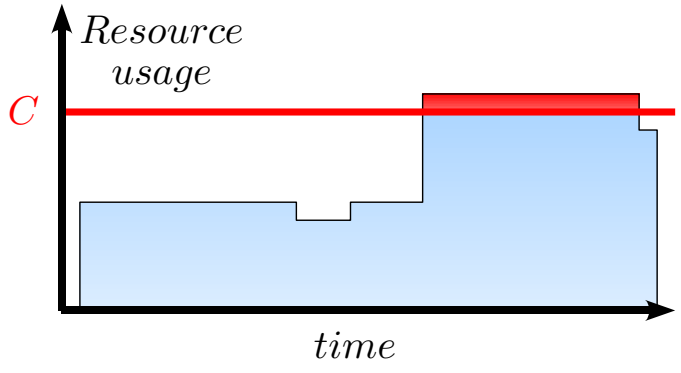
Minimization of resource overuse

- m parallel dedicated machines
- One renewable resource with capacity C
- Set of jobs to be executed on machine i: N_i
- Parameters of job T_j:
  - processing time (p_j)
  - time window (e_j, d_j)
  - resource demand (b_j)

Objective function

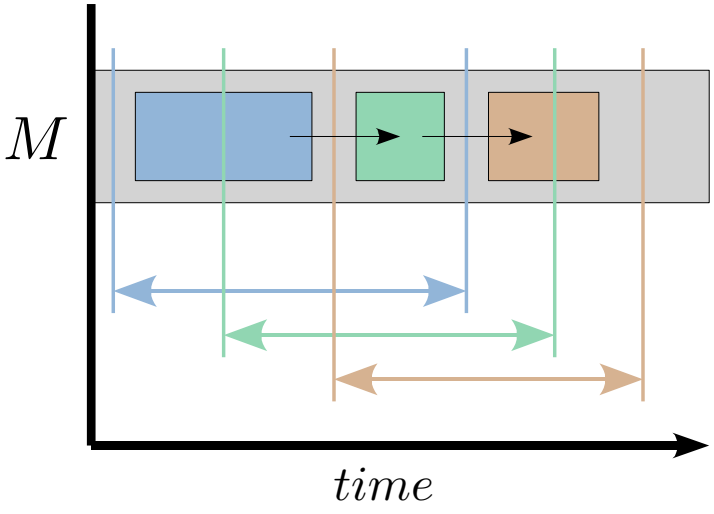
- Minimize total resource usage above C
- Minimize total squared resource usage above C
- Minimize quadratic deviation from mean resource demand (resource levelling)

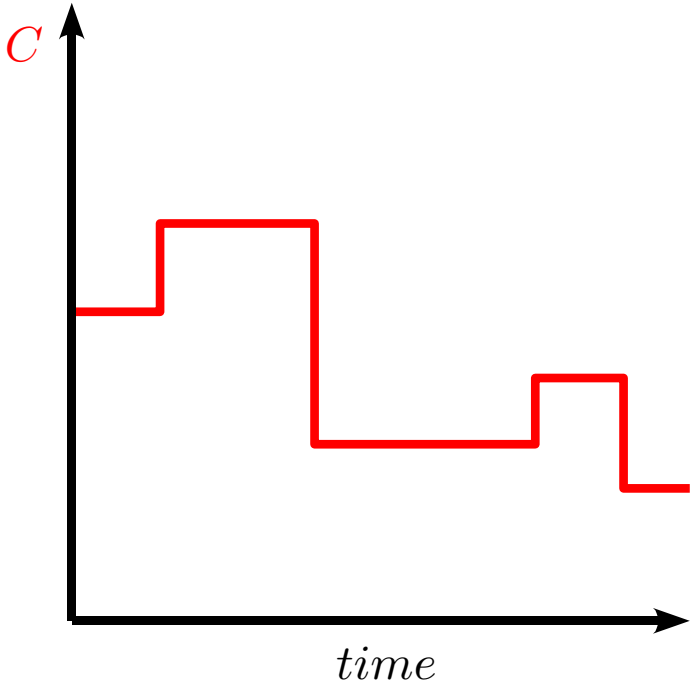




Complexity

- NP-hard in the strong sense
- One machine, variable resource availability: NP-hard in the strong sense
- One machine, variable resource availability, fixed sequence: polynomial





Formulation as an integer program

$$\min \sum_{t=0}^{D} y_t$$

subject to

$$\begin{aligned}
\sum_{t=0}^{D} x_{jt} &= 1, \quad \forall T_j \in N, \\
\sum_{T_j \in N_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} &\leq 1, \quad t \in \{0, \ldots, D\}, \ \forall i \in M, \\
\sum_{T_j \in N} \sum_{\tau=t-p_j+1}^{t} b_j x_{j\tau} &\leq C + y_t, \quad t \in \{0, \ldots, D\} \\
x_{jt} \in \{0, 1\} \ \forall T_j, t & \qquad y_t \geq 0 \ \forall t
\end{aligned}$$

Lagrangean relaxation

Original problem

$$\min \mathbf{cx}$$

subject to

$$\begin{aligned} \mathbf{Ax} &\leq \mathbf{b} \\ \mathbf{Dx} &\leq \mathbf{e} \\ \mathbf{x} &\geq 0, \mathbf{x} \in \mathbb{Z}^r \end{aligned}$$

Original problem

$$\max_{\boldsymbol{\lambda} \geq 0} \min \mathbf{cx} + \boldsymbol{\lambda}(\mathbf{Ax} - \mathbf{b})$$

subject to

$$\begin{aligned} \mathbf{Dx} &\leq \mathbf{e} \\ \mathbf{x} &\geq 0, \mathbf{x} \in \mathbb{Z}^r \end{aligned}$$

Lagrangean relaxation of the problem

$$\min \sum_{t=0}^{D} y_t$$

subject to

$$\begin{aligned}
\sum_{t=0}^{D} x_{jt} &= 1, \quad \forall T_j \in N, \\
\sum_{T_j \in N_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} &\leq 1, \quad t \in \{0, \ldots, D\}, \ \forall i \in M, \\
\sum_{T_j \in N} \sum_{\tau=t-p_j+1}^{t} b_j x_{j\tau} &\leq C + y_t, \quad t \in \{0, \ldots, D\} \\
x_{jt} \in \{0, 1\} \ \forall j, t \qquad & \qquad y_t \geq 0 \ \forall t
\end{aligned}$$

Lagrangean relaxation of the problem

$$\max_{\boldsymbol{\lambda} \geq 0} LB(\boldsymbol{\lambda}), \ \boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_D)$$

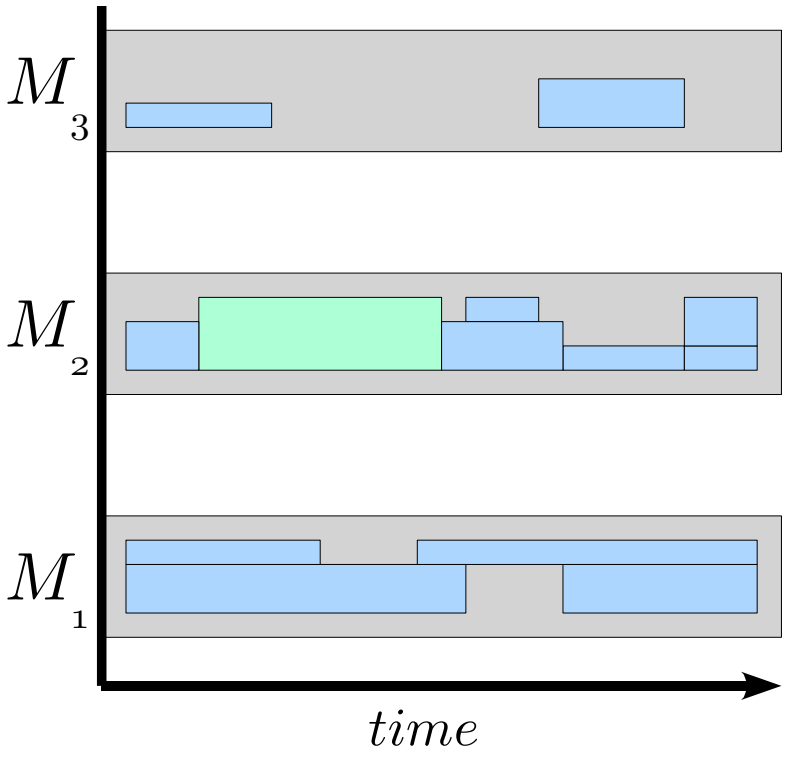
$$\begin{aligned} LB(\boldsymbol{\lambda}) &= \min \sum_{t=0}^{D} ((1-\lambda_t) y_t - \lambda_t C) \\ &\quad + \sum_{i \in M} \sum_{T_j \in N_i} \sum_{t=0}^{D} \sum_{\tau=t-p_j+1}^{t} \lambda_t b_j x_{j\tau} \end{aligned}$$

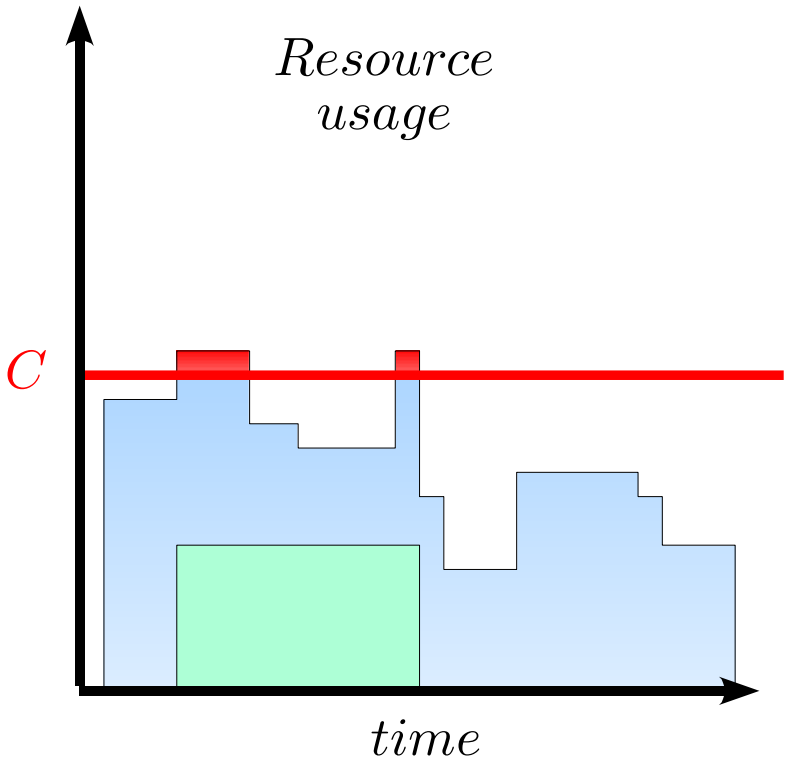
subject to

$$\begin{aligned} \sum_{t=0}^{D} x_{jt} &= 1, \quad \forall i \in M, \ \forall j \in N_i, \ x_{jt} \geq 0 \ \forall T_j, t \\ \sum_{T_j \in N_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} &\leq 1, \quad t \in \{0, \ldots, D\}, \ \forall i \in M \end{aligned}$$

Shaving

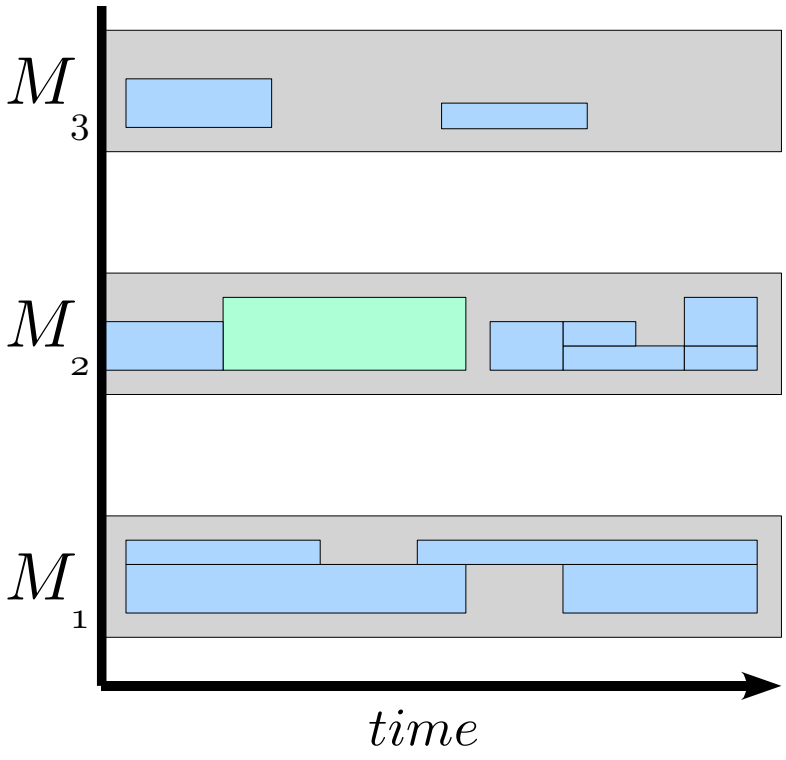
- $\boldsymbol{\lambda}$ is fixed
- Lower bound if T_j is started at time t: $\tilde{LB}_{jt}$

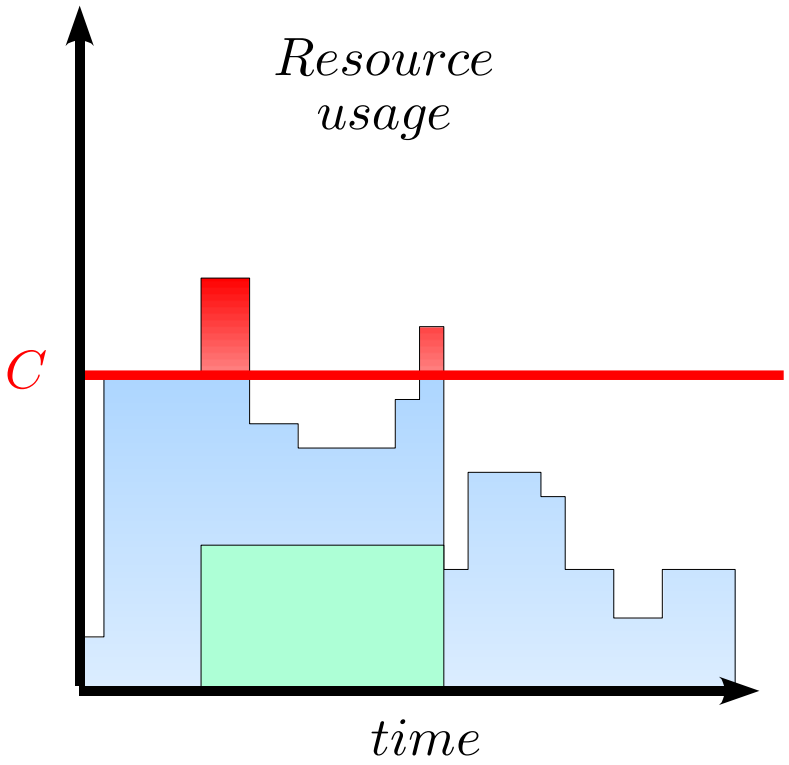




Shaving

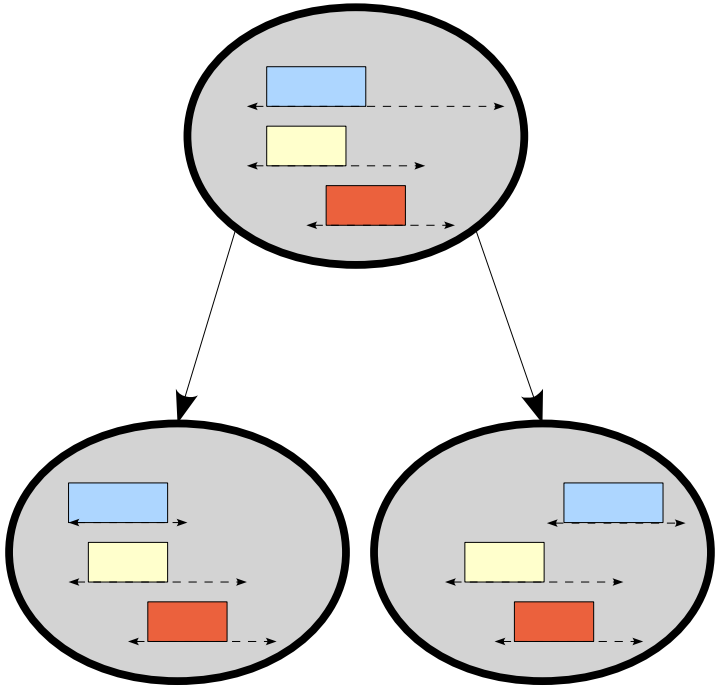
- $\boldsymbol{\lambda}$ is fixed
- Lower bound if T_j is started at time t: $\tilde{LB}_{jt}$





Branch&Bound

1. Constraint propagation
2. Calculation of lower bound
3. Shaving
4. Calculation of upper bound
5. Branching



Branch&Bound

	data used	**result**
Constraint propagation		narrowed time windows
Calculation of lower bound	λ in the parent	lower bound λ
Shaving	λ	improved lower bound narrowed time windows
Calculation of upper bound	solutions of LP-s	upper bound schedule
Branching	shaving	lower bound of children

Quadratic objective function

$$\min \sum_{t=0}^{D} y_t^2$$

subject to

$$\begin{aligned}
\sum_{t=0}^{D} x_{jt} &= 1, \quad \forall T_j \in N, \\
\sum_{T_j \in N_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} &\leq 1, \quad t \in \{0, \ldots, D\}, \ \forall i \in M, \\
\sum_{T_j \in N} \sum_{\tau=t-p_j+1}^{t} b_j x_{j\tau} &\leq C + y_t, \quad t \in \{0, \ldots, D\} \\
x_{jt} \in \{0, 1\} \ \forall T_j, t & \qquad y_t \geq 0 \ \forall t
\end{aligned}$$

Multiple renewable resources

$$\min \sum_{\rho=1}^{R} \sum_{t=0}^{D} w_\rho y_{\rho t} \quad \text{vagy} \quad \min \sum_{\rho=1}^{R} \sum_{t=0}^{D} w_\rho y_{\rho t}^2$$

subject to

$$\begin{aligned}
\sum_{t=0}^{D} x_{jt} &= 1, \quad \forall T_j \in N, \\
\sum_{T_j \in N_i} \sum_{\tau=t-p_j+1}^{t} x_{j\tau} &\leq 1, \quad t \in \{0, \ldots, D\}, \ \forall i \in M, \\
\sum_{T_j \in N} \sum_{\tau=t-p_j+1}^{t} b_j x_{j\tau} &\leq C_\rho + y_{\rho t}, \quad t \in \{0, \ldots, D\}, \ \rho \in \{1, \ldots, R\} \\
x_{jt} \in \{0, 1\} \ \forall T_j, t & \qquad y_t \geq 0 \ \forall t
\end{aligned}$$

Test results

- Generated test instances
- Resource limit:
  - Linear objective function: mean resource requirement
  - Qudratic objective function: 0 (resource levelling)
- 3 resources, with weights 4:3:2
- Initital solution can be found easily
- Runtime limit: 30 minutes

Linear objective function

	m5	m10	m20	avg
t10	6.26%	0.67%	0.34%	**2.42%**
t15	13.28%	5.94%	0.32%	**6.51%**
t20	18.29%	5.53%	2.23%	**8.69%**
avg	**12.61%**	**4.05%**	**0.97%**	**5.87%**

Branch&Bound

	m5	m10	m20	avg
t10	3.10%	0.24%	0.37%	**1.24%**
t15	11.28%	5.96%	1.62%	**6.29%**
t20	17.15%	7.48%	10.39%	**11.67%**
avg	**10.51%**	**4.56%**	**4.13%**	**6.40%**

ILOG CPLEX

mx indicates that the number of machines is x,
ty means that the number of tasks **one each machine** is y, so there are $x \cdot y$ tasks in the corresponding instances

Quadratic objective function

	m5	**m10**	**m20**	**avg**
t10	3.55%	1.10%	0.28%	**1.65%**
t15	4.22%	1.92%	1.14%	**2.43%**
t20	3.36%	1.29%	0.56%	**1.73%**
avg	**3.71%**	**1.44%**	**0.66%**	**1.94%**

Branch&Bound

	m5	**m10**	**m20**	**avg**
t10	1.51%	–	–	–
t15	–	–	–	–
t20	–	–	–	–
avg	–	–	–	–

ILOG CPLEX

mx indicates that the number of machines is x,
ty means that the number of tasks **one each machine** is y, so there are $x \cdot y$ tasks in the corresponding instances

Speed up using parallel computer[1]

		t10	t15	t20	avg
m10	**thr5**	2.65	3.09	3.29	*3.01*
	thr10	3.18	4.27	4.81	*4.09*
m20	**thr5**	2.58	3.17	3.33	*3.03*
	thr10	3.24	4.37	4.92	*4.18*
avg	**thr5**	*2.62*	*3.13*	*3.31*	***3.02***
	thr10	*3.21*	*4.32*	*4.87*	***4.13***

$$\frac{\text{CPU-time}}{\text{wall time}}$$

mx indicates that the number of machines is x,
ty means that the number of tasks **one each machine** is y, so there are $x \cdot y$ tasks in the corresponding instances
thr$z$ indicates that the number of processor threads is z

[1] Tested on the Sun Fire 15000 supercomputer of the National Information Infrastructure Development Institute of Hungary

Possible usage of other objective functions

- The lower bound can be calculated similarly for any objective function $f(\mathbf{y})$, for which

$$\min_{\mathbf{y}_l \leq \mathbf{y} \leq \mathbf{y}_u} f(\mathbf{y}) - \boldsymbol{\lambda}\mathbf{y}$$

  can be solved "easily"
- The heuristic used for calculating the upper bound can be applied – and is at least pseudopolynomial – for any objective function $f(\mathbf{y})$ that can be formulated as

$$f(\mathbf{y}) = \sum_{t=0}^{D} f_t(\mathbf{y_t})$$

Publication

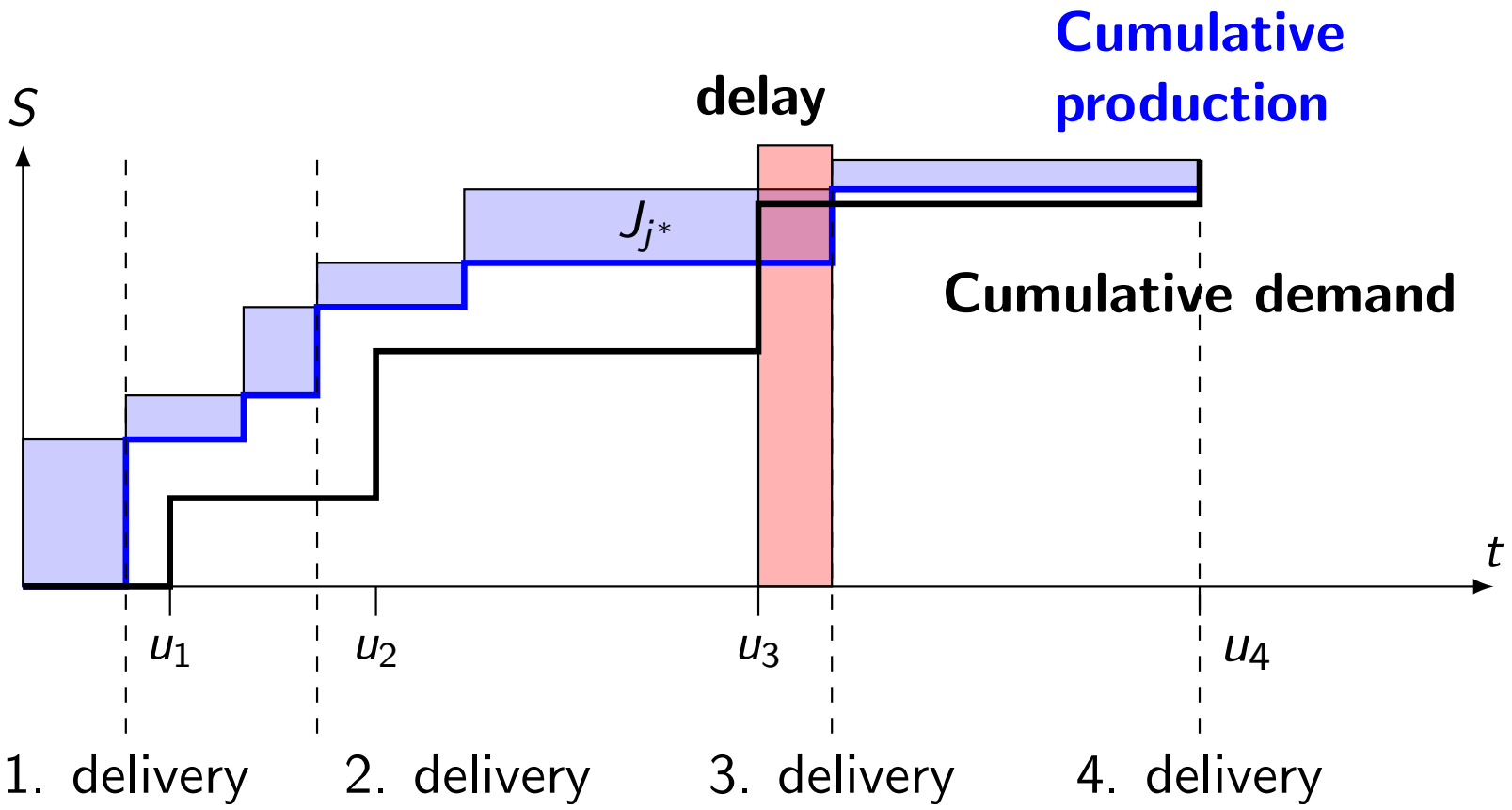
- Drótos, M., & Kis, T. (2011). Resource leveling in a machine environment. European Journal of Operational Research, 212(1), 12-21.

Machine scheduling with non-renewable resources

joint work with Péter Györgyi and Márton Drótos

Outline

Producer jobs



Problem description

Problem data:

- Single machine
- Set of product types $\mathcal{R}$.
- Set of jobs $\mathcal{J}$ with processing times p_j, and a quantity a_{ij} produced from each product type $i \in \mathcal{R}$.
- External demand requests with deadlines $u_1 \leq \cdots \leq u_q$, and quantities requested $b_{i,\ell}$, $(i \in \mathcal{R})$.
- Assumptions: $\sum_j a_j = \sum_\ell b_\ell$ and $u_q = \sum p_j$.

Schedules:

- Starting time S_j for each job J_j,
- Feasible if and only if jobs do not overlap in time
- Delivery time of the ℓth external demand request is the earliest time point when the total amount produced from product i equals or exceeds $\sum_{k=1}^{\ell} b_{i,k}$, $i \in \mathcal{R}$

Parallel machines and consumer jobs

- We have m parallel machines ($\mathcal{M}$), a set of jobs $\mathcal{J}$ and a set of resources $\mathcal{R}$.
- Preemption is not allowed.
- Each job has a processing time and some resource requirements, i.e. we can schedule it only if there are enough resources to cover its requirements.
- These resources are consumed by the jobs at the time of starting.
- Each resource has an initial stock, which is replenished in known quantities at given dates.
- Objective functions: $C_{\max}$, $L_{\max}$, $\sum w_j C_j$.

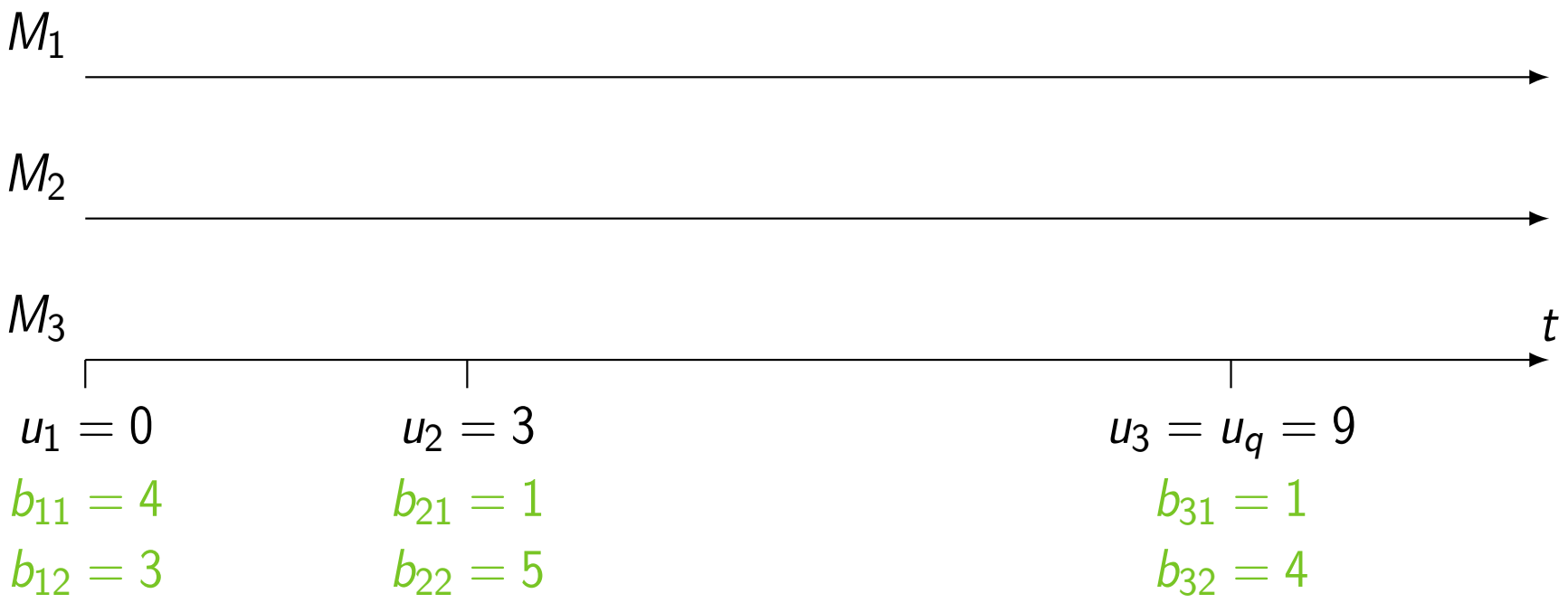
Parallel machines and consumer jobs (cntd.)

- The problem was introduced by Carlier (1984).
- The problem has many practical application (production scheduling, logistics).
- Main goal: approximability results for different sub-problems.
- We have to decide whether there is a PTAS or an FPTAS for different variants of the problem.
- Notations: n jobs, m machines, rm number of resources, q supplies $(0 = u_1 < u_2 < \ldots < u_q)$.
- $b_{\ell,i}$: amount of the resource supplied at u_ℓ from resource i.
- $a_{i,j}$: requirement of job J_j from resource i

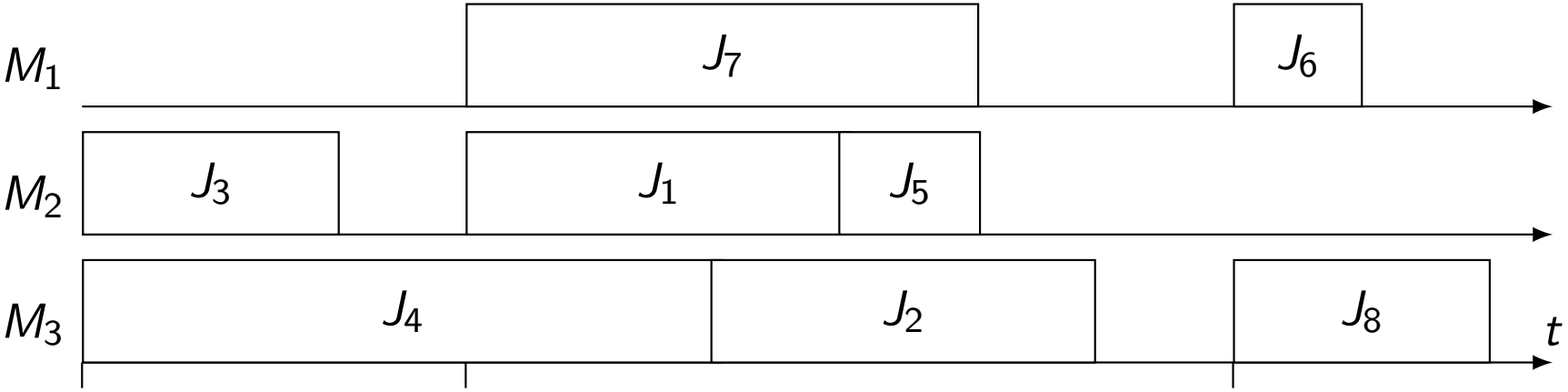
Consumer jobs

3 machines, 2 resources, 3 supplies and 8 jobs

p_j	3	3	2	5	1	1	4	2
a_{1j}	0	1	2	1	0	1	1	0
a_{2j}	1	2	1	2	1	0	1	4



A feasible schedule

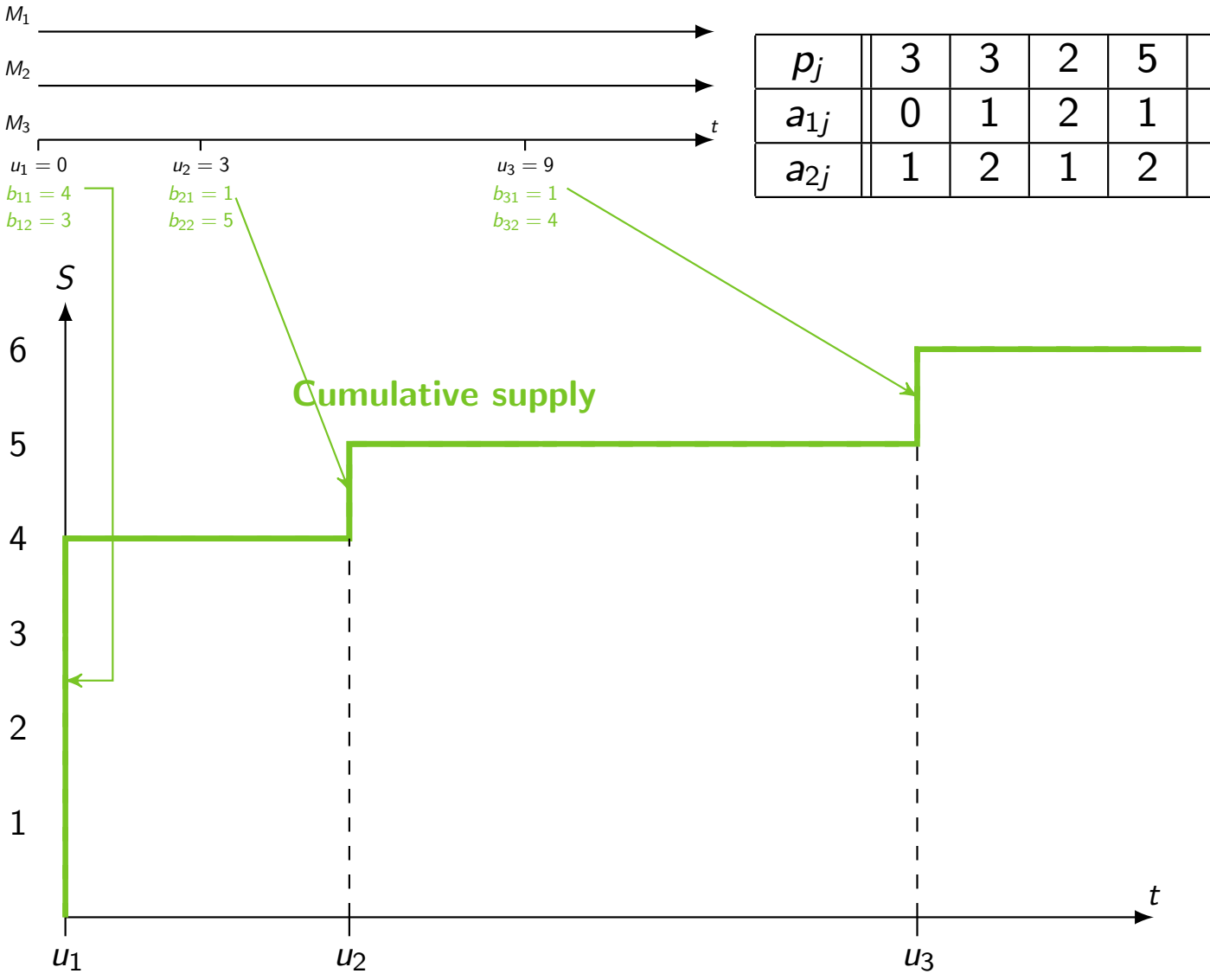


$u_1 = 0$ $\quad$ $u_2 = 3$ $\quad$ $u_3 = u_q = 9$

$b_{11} = 4$ $\quad$ $b_{21} = 1$ $\quad$ $b_{31} = 1$

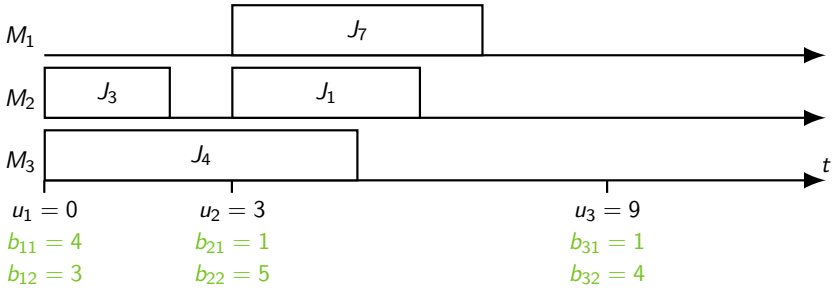
$b_{12} = 3$ $\quad$ $b_{22} = 5$ $\quad$ $b_{32} = 4$

Supplies and consumption (resource 1)

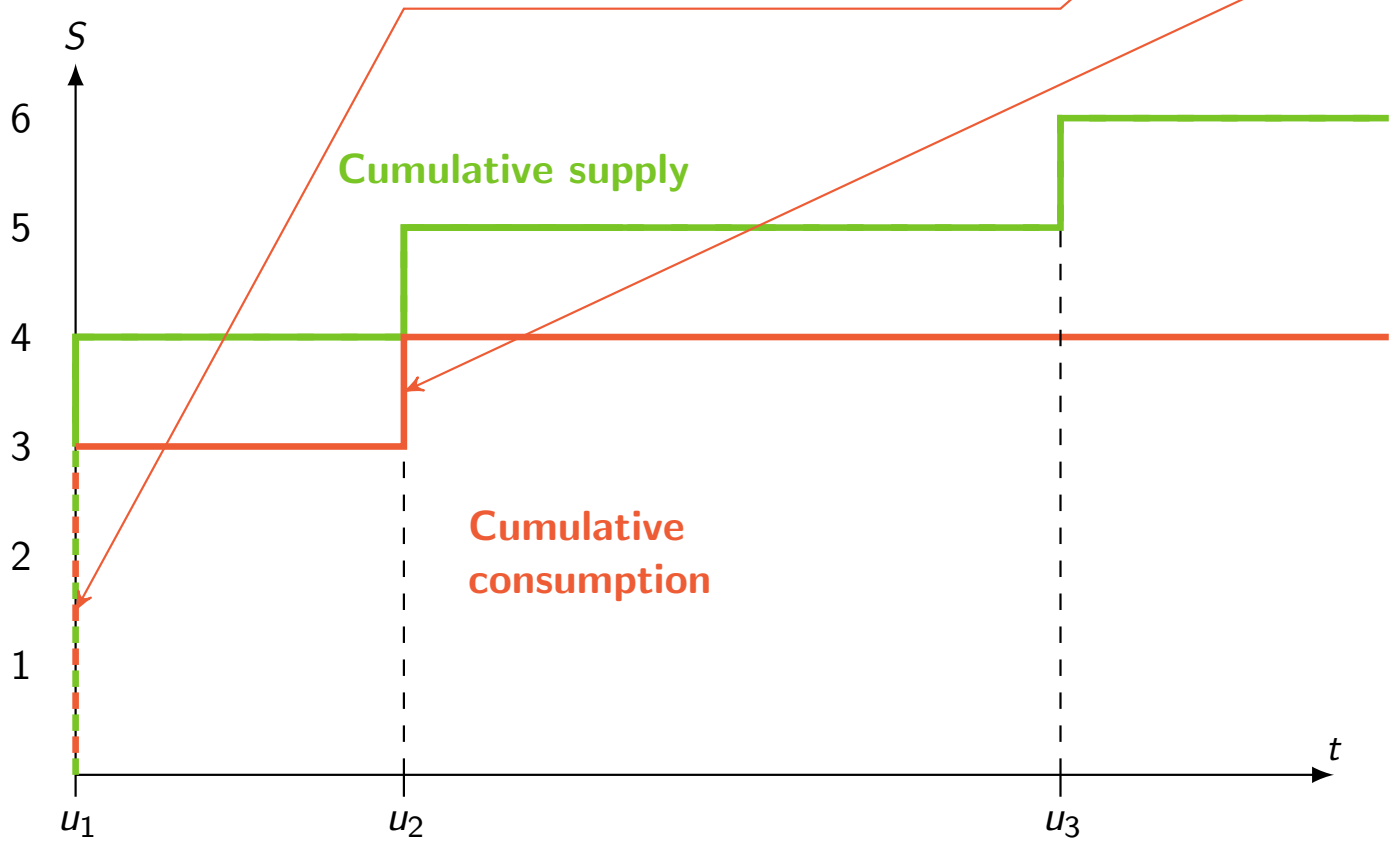


p_j	3	3	2	5	1	1	4	2
a_{1j}	0	1	2	1	0	1	1	0
a_{2j}	1	2	1	2	1	0	1	4

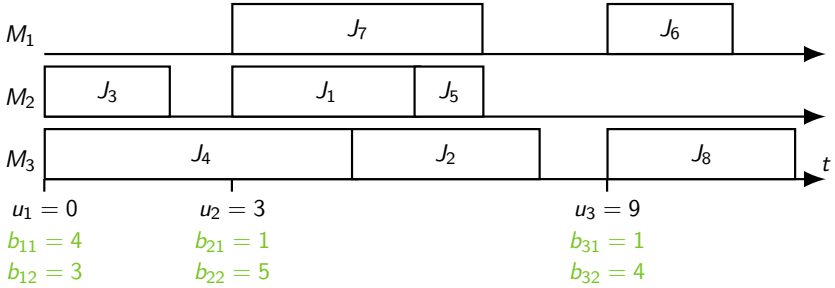
Supplies and consumption (resource 1)



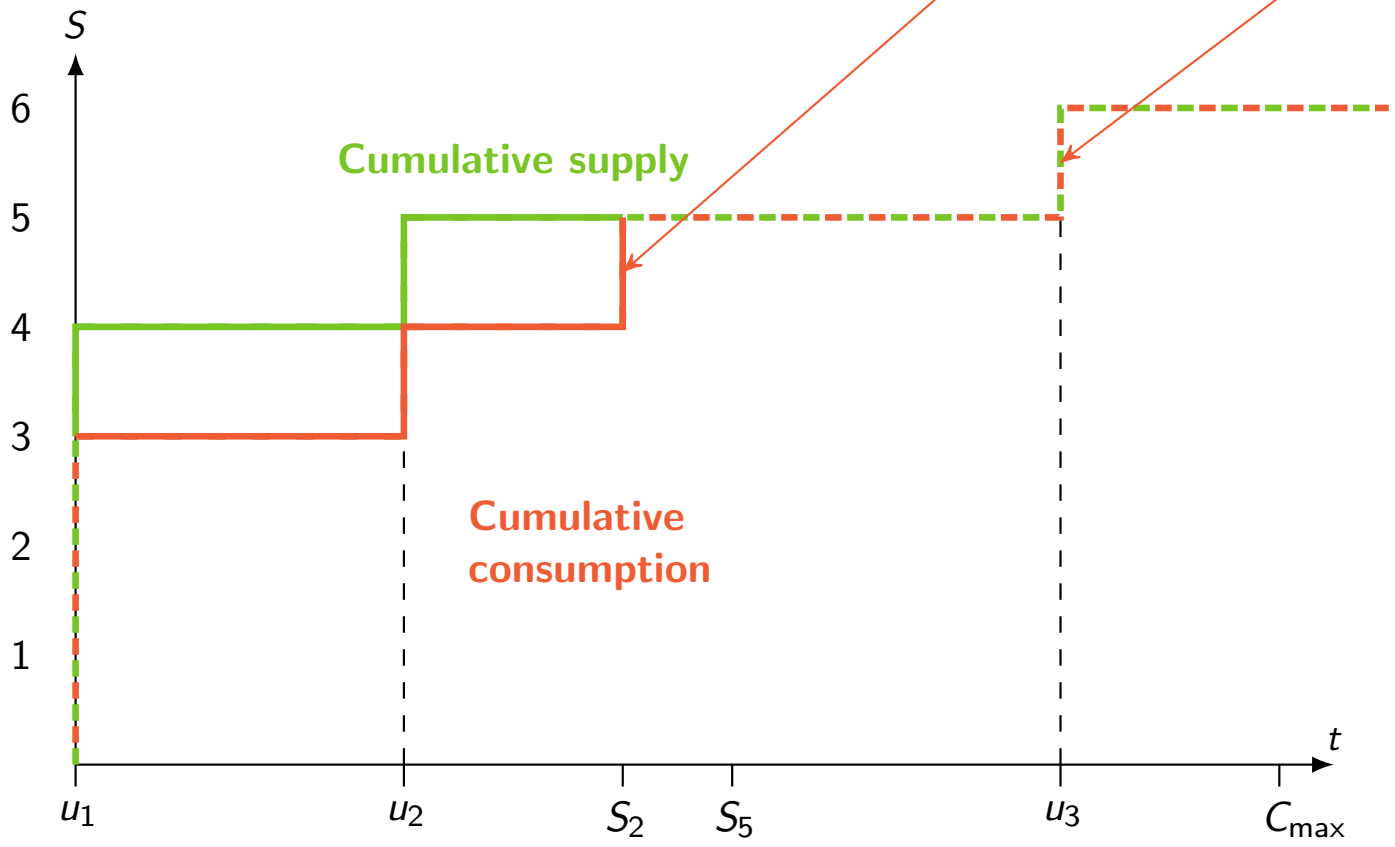
p_j	3	3	2	5	1	1	4	2
a_{1j}	0	1	2	1	0	1	1	0
a_{2j}	1	2	1	2	1	0	1	4



Supplies and consumption (resource 1)



p_j	3	3	2	5	1	1	4	2
a_{1j}	0	1	2	1	0	1	1	0
a_{2j}	1	2	1	2	1	0	1	4



Some easy special cases of the delivery tardiness problem

Let γ be a non-decreasing function of the delivery times.

1. '$1|\ p_j = p, |\mathcal{R}| = 1\ |\ \gamma$' is solvable in $O(n \log n)$ time
   - Schedule the jobs in non-increasing $a_{1,j}$ order.
2. '$1|\ a_j = a\ |\ \gamma$' is solvable in $O(n \log n)$ time
   - Schedule the jobs in non-decreasing processing time order (SPT rule)

Reductions between optimization problems

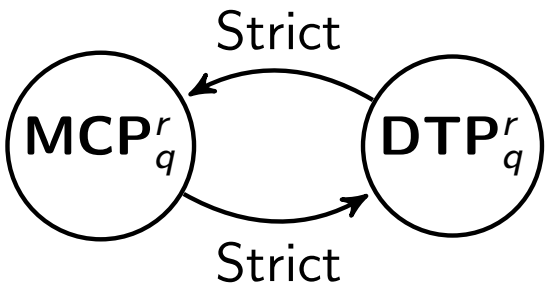
Reduction from problem class Π_1 to problems class Π_2 is a pair of functions f and g where f transforms instances of Π_1 to that of Π_2, and g transforms solutions of Π_2 to that of Π_1:

$$\begin{array}{lccc}
\text{Problems:} & \Pi_1 & & \Pi_2 \\
\hline
\text{Instances:} & I_1 & \xrightarrow{f} & f(I_1) \\
 & & & \downarrow \\
\text{Solutions:} & g(I_1, y) & \xleftarrow{g} & y
\end{array}$$

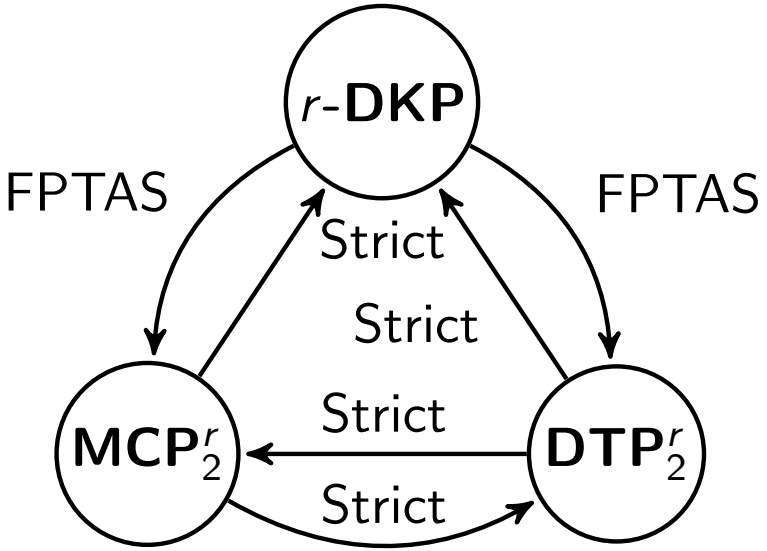
1. L-reduction: preserves approximation ratio with a constant factor
2. Strict reduction: preserves approximation ratio
3. PTAS reduction: preserves PTAS
4. FPTAS reduction: preserves FPTAS

Reductions

Summary of approximation preserving reductions between scheduling and knapsack problems



q and r are arbitrary



$q = 2$, r arbitrary

Equivalence of the delivery tardiness-, and the material consumption problems

Lemma

Let $I = \{n, q, (p_j, a_j)_{j=1}^{n}, (u_\ell, b_\ell)_{\ell=1}^{q}\}$ *be an instance of the Delivery tardiness problem. Define an instance* $I' = \{n, q, (p_j, a_j)_{j=1}^{n}, (u'_\ell, b'_\ell)_{\ell=1}^{q}\}$ *of the Material consumption problem:*

$$\begin{aligned} u'_\ell &= u_q - u_{q+1-\ell} \\ b'_\ell &= b_{q+1-\ell} \end{aligned} \qquad \ell = 1, \ldots, q.$$

Then, if σ *is a sequence of jobs with maximum delivery tardiness of* $T^{\sigma}_{\max}$ *for* I*, then scheduling the jobs in reverse* σ *order gives a schedule of makespan* $u_q + T^{\sigma}_{\max}$ *for instance* I' *of the Material consumption problem.*

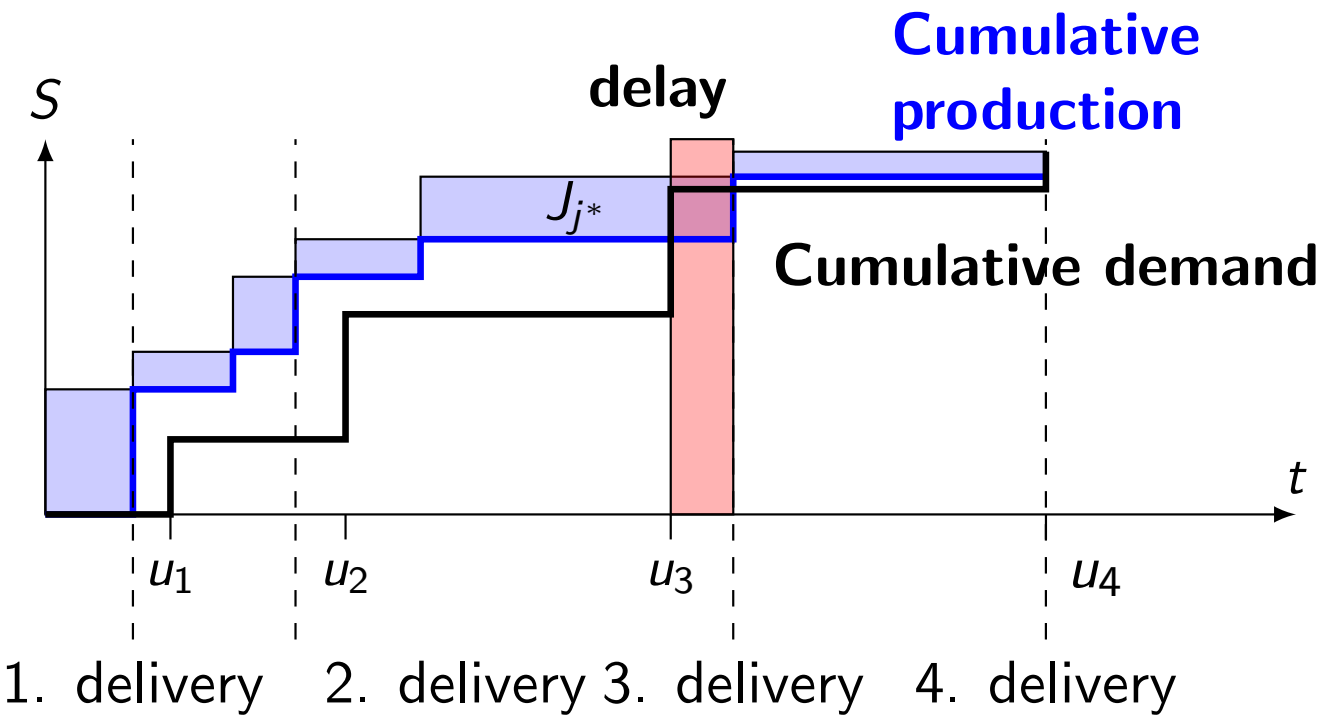
Equivalence (part 2)

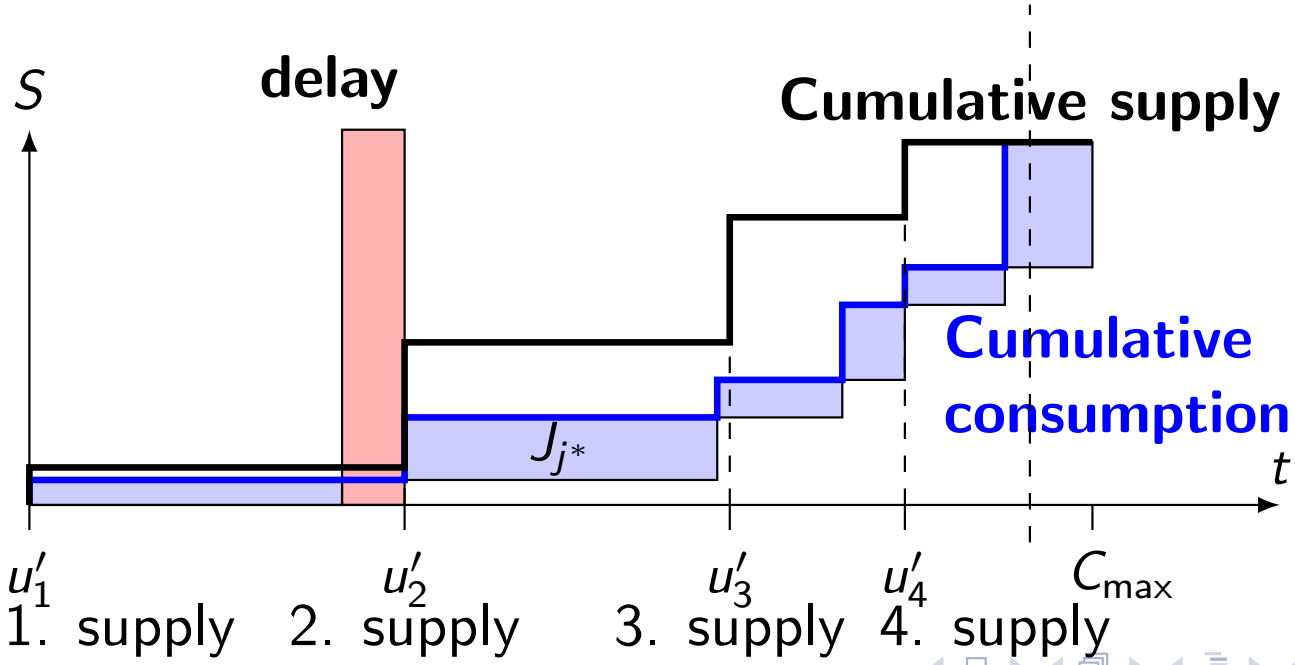
Lemma

Given an instance $I = \{n, q, (p_j, a_j)_{j=1}^n, (u_\ell, b_\ell)_{\ell=1}^q\}$ *of the Material consumption problem. Define an instance* $I' = \{n, q, (p_j, a_j)_{j=1}^n, (u'_\ell, b'_\ell)_{\ell=1}^q\}$ *of the Delivery tardiness problem:*

$$\begin{aligned} u'_\ell &= u_q - u_{q+1-\ell} \\ b'_\ell &= b_{q+1-\ell} \end{aligned} \qquad \ell = 1, \ldots, q.$$

Then, if S *is a schedule with a makespan of* $C_{\max}^S$ *for* I*, then scheduling the jobs in reverse order (without any delays among them) gives a schedule of maximum delivery tardiness at most* $C_{\max}^S - u_q$ *for instance* I' *of the Delivery tardiness problem.*


S
delay
Cumulative production
J_{j^*}
Cumulative demand
t
u_1
u_2
u_3
u_4
1. delivery
2. delivery
3. delivery
4. delivery


S
delay
Cumulative supply
Cumulative consumption
J_{j^*}
t
u'_1
u'_2
u'_3
u'_4
C_{max}
1. supply
2. supply
3. supply
4. supply

Consequences of the equivalence

Corollary

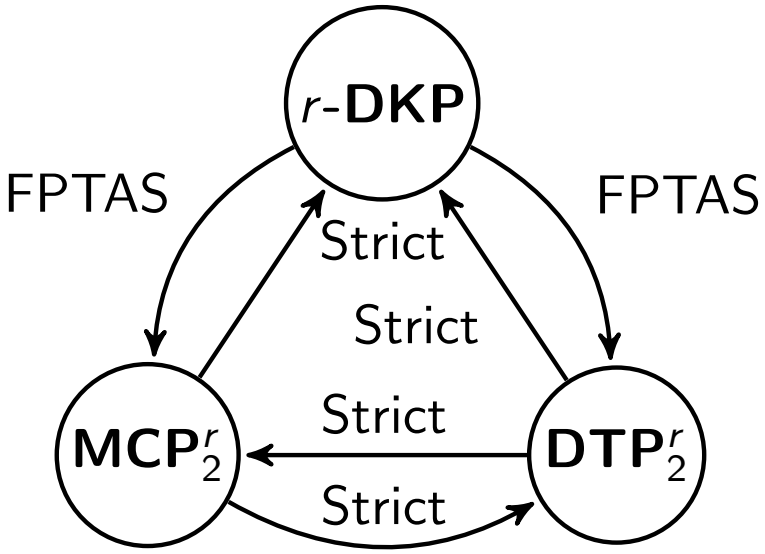
Let (I_M, I_D) be corresponding instances of the Material consumption- and the Delivery tardiness problems. Then the optimum value $C^*_{\max}(I_M)$ of the Material consumption problem equals $u_q(I_D) + T^*_{\max}(I_D)$.

In order to approximate the Delivery tardiness problem, use the shifted tardiness objective: $(u_q - u_1) + T_{\max}$

Theorem

The Delivery tardiness problem admits a FPTAS / PTAS for some q and r if and only if the Material consumption problem admits an FPTAS / PTAS for the same q and r.

Consequences of the reductions



Theorem

Any $1/(1-\varepsilon)-$approx. algorithm for $r-$DKP yields an $(1+\varepsilon)-$approx. algorithm for MCP_2^r and DTP_2^r.

Theorem

There is no FPTAS for MCP_2^r and DTP_2^r for any fixed $r \geq 2$, unless $P = NP$.

Approximability results

Obj.	#Machines m	#Supplies q	#Resources rm	Release dates r_j	PTAS	FPTAS
$C_{\max}$	1	2	1	no	yes	yes
	1	2	1	yes	yes	?
	1	2	$const. \geq 2$	yes/no	yes	no
	1	2	arbitrary	yes/no	no	no
	1	$const. \geq 3$	1	yes/no	yes	?
	1	$const. \geq 3$	$const. \geq 2$	yes/no	yes	no
	1	arbitrary	1*	yes/no	yes	no
	$const \geq 2$ +ddc. jobs**	arbitrary	$const. \geq 2$	yes/no	yes yes	no
	arbitrary	2	2	yes/no	no	no
	arbitrary	arbitrary	1	yes/no	yes	no
$L'_{\max}$	$const$	arbitrary	1*	no	yes	?
$\sum w_j C_j$	1	2	1	no	yes	yes

* under the condition $a_j = \lambda p_j$ ** even if only a $\mathcal{J}' \subseteq \mathcal{J}$ subset of jobs is dedicated

Publications

- Drótos, M., & Kis, T. (2013). Scheduling of inventory releasing jobs to minimize a regular objective function of delivery times. Journal of Scheduling, 16(3), 337-346.
- Györgyi, P., & Kis, T. (2014). Approximation schemes for single machine scheduling with non-renewable resource constraints. Journal of Scheduling, 17(2), 135-144.
- Györgyi, P., & Kis, T. (2015). Reductions between scheduling problems with non-renewable resources and knapsack problems. Theoretical Computer Science, 565, 63-76.
- Györgyi, P., & Kis, T. (2015). Approximability of scheduling problems with resource consuming jobs. Annals of Operations Research, 235(1), 319-336.
- Kis, T. (2015). Approximability of total weighted completion time with resource consuming jobs. Operations Research Letters, 43(6), 595-598.
- Györgyi, P., & Kis, T. (2015). Approximation schemes for parallel machine scheduling with non-renewable resources. European J Operational Res., in press.

Solving resource constrained shortest path problems with LP-based methods

joint work with Markó Horváth

The Resource Constrained Shortest Path Problem (RCSPP)

Given:

- $D = (V, A)$ directed graph, $s \neq t$ selected nodes
- $c : A \to \mathbb{Z}$ cost function
- $w^r : A \to \mathbb{Z}$ $(r = 1, \ldots, m)$ resource consumptions
- $W^r \in \mathbb{Z}$ $(r = 1, \ldots, m)$ resource limits

Goal: Find a minimal cost s–t path respecting the resource limits:

$$\min_{P \in \mathcal{P}_{st}} \left\{ \sum_{a \in P} c_a \ : \ \sum_{a \in P} w_a^r \leq W^r \text{ for all } r = 1, \ldots, m \right\}$$

Assumption: D contains no directed cycle of negative total cost, or of negative total resource consumption for any of the resource types

Motivations

The RCSPP (and its variants) has several application fields in practice, and naturally arises as auxiliary problem in column generation schemes

- Crew scheduling, Integrated vehicle and crew scheduling
- Crew rostering
- Fleet management
- Air cargo planning and routing
- ...

Aims and results

- LP-based branch-and-bound procedure
  - Variable fixing method
  - Primal heuristic
  - Valid inequalities (cutting planes)
- Thorough computational results
  - Evaluation of the components
  - Comparison with state-of-the-art methods

Integer programming formulation

The RCSPP can be formulated as:

$$\min \sum_{a \in A} c_a x_a \tag{1}$$

$$\sum_{a \in \delta^-(i)} x_a - \sum_{a \in \delta^+(i)} x_a = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall\ i \in V \tag{2}$$

$$\sum_{a \in A} w_a^r x_a \leq W_a^r \qquad \forall\ r = 1, \ldots, m \tag{3}$$

$$x_a \in \{0, 1\} \qquad \forall\ a \in A \tag{4}$$

Dumitrescu-Boland preprocessing scheme

Preprocessing scheme of Dumitrescu and Boland (2003):

Input:

- Original graph D
- Nodes s and t, functions c and w^r $(r = 1, \ldots, m)$
- Current upper bound U

Process (repeatedly):

- Remove arcs (and nodes) if they cannot appear in an optimal solution
- Feasible path is found $\Rightarrow$ update U

Termination:

- Optimal solution is found $\Rightarrow$ STOP (problem is solved)
- Problem is proved to be infeasible $\Rightarrow$ STOP (problem is solved)
- No more arcs/nodes can be removed

Computational results — Dumitrescu-Boland preprocessing scheme

Table: Properties of well-known instance sets

	Set	Ins	Res	Nodes		Arcs	
				Min	Max	Min	Max
I1	Beasley	12	1	100	500	955	4 978
I2	Beasley	12	10	100	500	990	4 868
D1	Dumitrescu & Boland	8	1	10 002	135 002	29 900	404 850
D2	Dumitrescu & Boland	56	1	625	40 000	2 400	159 200
S	Santos	880	1	10 000	40 000	15 000	800 000

Table: Summary of experiments on well-known instance sets

	Set	Ins	Preprocessing		MIP	
			Time	Solved	BBN	Time
I1	Beasly	**12**	0.01	**12**	–	–
I2	Beasly	**12**	0.03	**10**	1.0	0.45
D1	Dumitrescu & Boland	8	0.38	0	1.0	179.28
D2	Dumitrescu & Boland	56	0.12	1	3.0	22.96
S	Santos	**880**	0.12	**878**	2.0	160.71

Table: Summary of experiments on instance set S (group 1)

Set	RPM[1]	PA[2]	LPB[3]	Set	RPM	PA	LPB	Set	RPM	PA	LPB
S1	0.88	0.01	0.01	S7	3.00	0.02	0.03	S13	13.39	0.05	0.06
S2	1.61	0.02	0.01	S8	5.73	0.03	0.04	S14	22.05	0.08	0.08
S3	4.20	0.02	0.02	S9	13.45	0.06	0.04	S15	53.39	0.12	0.15
S4	7.73	0.04	0.04	S10	28.84	0.11	0.09	S16	112.92	0.21	0.22
S5	11.38	0.07	0.06	S11	43.66	0.17	0.14	S17	167.53	0.29	0.36
S6	16.61	0.07	0.08	S12	64.85	0.17	0.22	S18	248.18	0.39	0.49

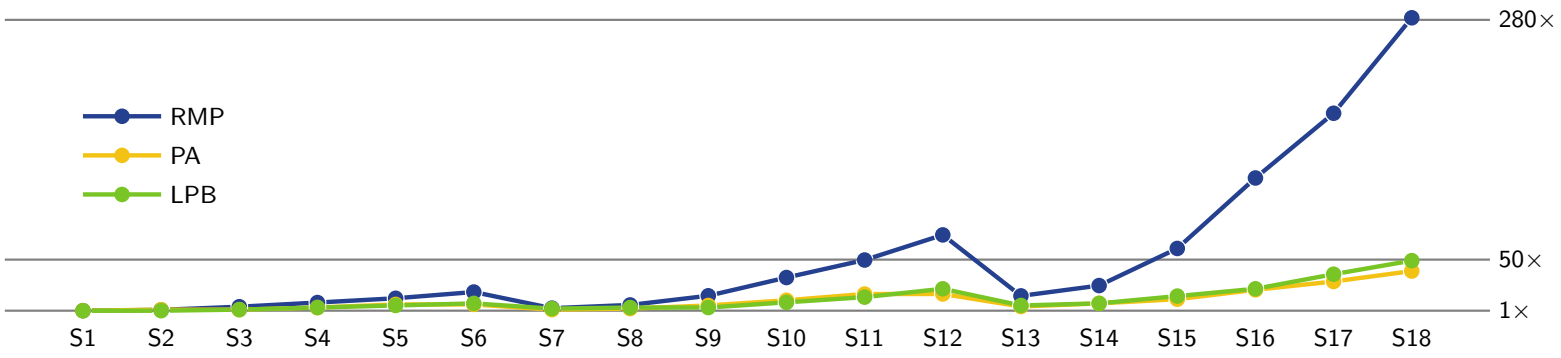


Figure: Execution times scaled to set S1

[1] Puglieze & Guerriero (2013): tested with Intel Core i7-620M, 2.67 GHz CPU, under Windows 7

[2] Lozano & Medaglia (2013): tested with Intel Core 2 Duo P8600, 2.4 GHz CPU, under Windows XP

[3] [?]: tested with Intel Core i7-4710MQ, 2.5 GHz CPU, under Windows 7

Variable fixing in search-tree nodes

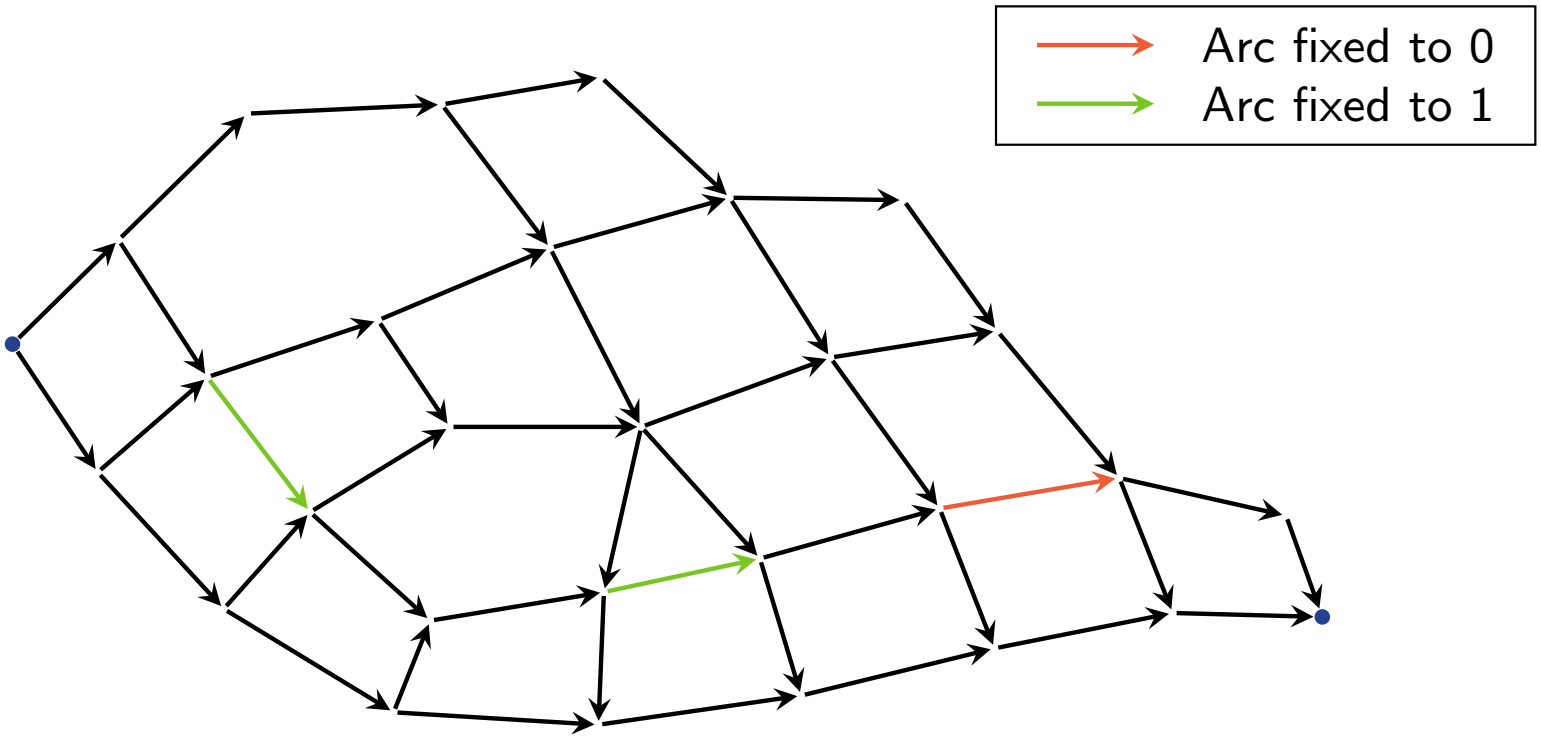


Figure: Fixed variables (arcs) in graph D

Variable fixing in search-tree nodes

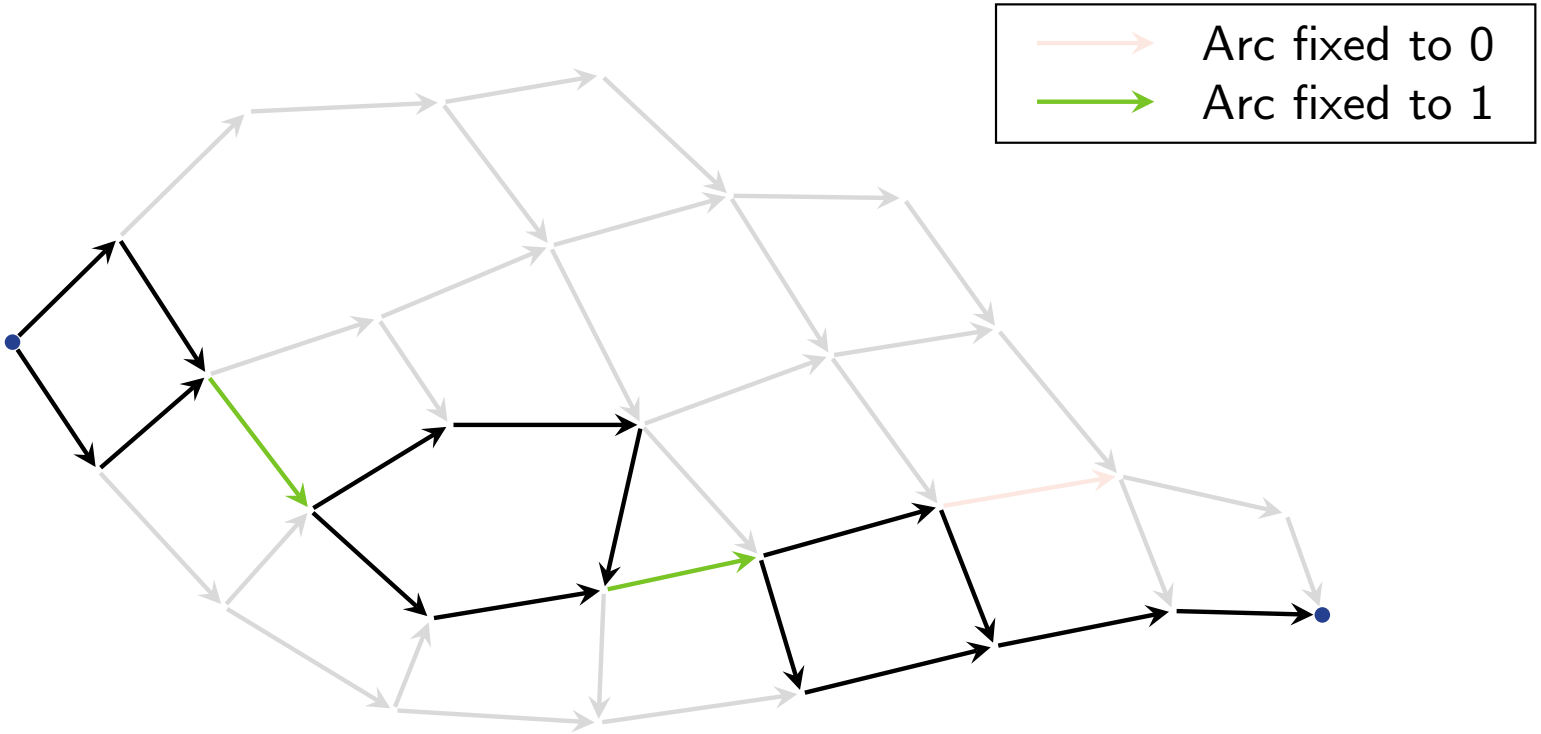


Figure: Subgraph $\hat{D}$ according to fixed variables

Variable fixing method of [?]:

Input:

- ~~Original graph D~~ Subgraph $\hat{D}$
- Nodes s and t, functions c and w^r $(r = 1, \ldots, m)$
- Current upper bound U = upper cutoff value

Process (repeatedly):

- ~~Remove arcs (and nodes)~~ Fix arcs to zero if they cannot appear in an (global) optimal solution
- Feasible path is found $\Rightarrow$ update U and the upper cutoff value

Termination:

- Optimal solution is found $\Rightarrow$ ~~STOP~~ Prune the node
- Problem is proved to be infeasible $\Rightarrow$ ~~STOP~~ Prune the node
- No more arcs~~/nodes can be removed~~ can be fixed

Computational results

Instance sets of Horváth and Kis (2016):

- Our randomly generated 240 instances can be grouped into 12 classes (G1-G12) according to their sizes, and the method used to generate their resource limits
- In all cases we used 10 resource functions

parameters of instance groups		number of nodes*		
		500	1000	1500
resource limits	reduced ($p = 20\%$)	G1	G5	G9
	uniform ($W = 20$)	G2	G6	G10
	uniform ($W = 30$)	G3	G7	G11
	uniform ($W = 40$)	G4	G8	G12

* a graph with n nodes contains approximately $10n$ arcs

Table: Summary of variable fixing experiments

	Set	BBN	Time	Set	BBN	Time	Set	BBN	Time
PLAIN	G1	1360.1	57.7	G5	1499.2	235.3	G9	2165.9	644.7
VARFIX		881.7	**19.0**		1000.3	**54.0**		1101.4	**93.5**
PLAIN	G2	1883.6	88.9	G6	2621.3	392.9	G10	2308.4	679.9
VARFIX		1351.3	**33.5**		1149.5	**63.5**		782.5	**71.0**
PLAIN	G3	4275.3	187.1	G7	4877.5	709.0	G11	5533.7	1400.7
VARFIX		3775.9	**84.5**		4281.7	**235.4**		4864.3	**432.6**
PLAIN	G4	3699.0	143.4	G8	4222.5	560.8	G12	6044.6	1434.0
VARFIX		3294.8	**63.9**		4620.6	**223.2**		5636.2	**469.2**

PLAIN: preprocessing + branch-and-bound
VARFIX: preprocessing + branch-and-bound with variable fixing

Primal heuristic

Once a fractional solution $\bar{x}$ to the LP-relaxation is found we perform a depth-first-search from node s on the support graph $\bar{D}$ of the solution

$$\bar{D} = \left(V, \bar{A} = \{a \in A \ : \ \bar{x}_a > 0\}\right)$$

- node labels $\lambda_s(u)$ and $\lambda_t(u)$: distance (for a fixed resource) from the source node and to the sink node, respectively.
- When we visit a node u we can store/update a label $\lambda_s(u)$ about the current s–u path
- When we backtrack to a node u we can store/update a label $\lambda_t(u)$ about an u–t path
- When for a node u both of the labels $\lambda_s(u)$ and $\lambda_t(u)$ exist we can check the feasibility of the appropriate s–t path

Computational results

Table: Summary of variable fixing and primal heuristic experiments

	Set	BBN	Time	Set	BBN	Time	Set	BBN	Time
PLAIN	G1	1360.1	57.7	G5	1499.2	235.3	G9	2165.9	644.7
HEURSOL		1191.3	49.6		1223.9	175.2		1841.0	501.1
VARFIX		881.7	**19.0**		1000.3	54.0		1101.4	**93.5**
ALL		775.3	19.4		877.8	**53.6**		1090.3	103.5
PLAIN	G2	1883.6	88.9	G6	2621.3	392.9	G10	2308.4	679.9
HEURSOL		1769.3	81.5		2303.7	339.2		2092.1	580.8
VARFIX		1351.3	33.5		1149.5	**63.5**		782.5	**71.0**
ALL		1265.9	**31.8**		1086.1	66.8		660.6	83.2
PLAIN	G3	4275.3	187.1	G7	4877.5	709.0	G11	5533.7	1400.7
HEURSOL		3261.0	134.2		4410.4	615.0		4637.2	1087.5
VARFIX		3775.9	84.5		4281.7	235.4		4864.3	432.6
ALL		3412.2	**76.6**		3722.3	**213.0**		3990.3	**370.9**
PLAIN	G4	3699.0	143.4	G8	4222.5	560.8	G12	6044.6	1434.0
HEURSOL		2742.5	101.0		3644.5	444.9		5150.7	1073.3
VARFIX		3294.8	63.9		4620.6	223.2		5636.2	469.2
ALL		3102.2	**62.4**		3982.5	**197.9**		5067.4	**432.3**

PLAIN: preprocessing + branch-and-bound
HEURSOL: preprocessing + branch-and-bound with primal heuristic
VARFIX: preprocessing + branch-and-bound with variable fixing
ALL: preprocessing + branch-and-bound with primal heuristic and variable fixing

Valid inequalities

We have generalized two classes of valid inequalities of Garcia (2009):

- s–t cut based inequalities
  - polynomial separation procedure
- Infeasible subpath based inequalities
  - NP-hard to separate
  - heuristic separation procedure
- Cuts were tested, they are significantly better on average than solver's built in cuts, but we can get even better results without generating any cuts, at least on the pure RCSPP instances generated by us.

Publication

- Horváth, M., & Kis, T. (2016). Solving resource constrained shortest path problems with LP-based methods. Computers & Operations Research, 73, 150-164.

Bilevel machine scheduling problems

joint work with András Kovács

Outline

- Bilevel optimization
- The bilevel total weighted completion time problem
- The bilevel order acceptance problem
- Open problems

Bilevel optimization

- Two players: leader and follower
- Leader's role
  - controls the variables x
  - decides first in view of the possible actions of the follower
  - aims to optimise an objective function which depends on x and on the Follower's response $\bar{y}$
- Follower's role
  - controls the variables y
  - decides second in view of the decision $\bar{x}$ of the Leader
  - aims to optimise an objective function which depends on y and possibly $\bar{x}$
- Leader's objective:
  - Optimistic: $\min_x f(x, y)$ subject to $(x, y) \in L$ and $y \in opt_F(x)$
  - Pessimistic: $\min_x \max_y f(x, y)$ subject to $(x, y) \in L$ and $y \in opt_F(x)$

The bilevel total weighted completion time problem

- n jobs and m parallel, identical machines, no preemption
- Leader: assigns jobs to machines ($J = J_1 \cup J_2 \cup \cdots \cup J_m$)
  - Optimistic objective: $\min \sum_{j \in J} w_j^1 C_j$
  - Pessimistic objective: $\min \max \sum_{j \in J} w_j^1 C_j$
- Follower: sequences the assigned jobs on each machine

$$\min \sum_{i=1}^{m} \sum_{j \in J_i} w_j^2 C_j$$

- For a given machine assignment $J_1, \ldots, J_m$, Follower solves m single machine problems $1 || \sum_j w_j^2 C_j$ by Smith's rule (WSPT order)
- Leader has to find the best assignment knowing the strategy of the Follower

Results on bilevel total weighted completion time problem

Restriction	**Complexity**
no restriction	decision version is NP-complete
$w^1 \equiv 1$, w^2 induces an increasing proc. time order	equivalent to $P\|\|\sum_j C_j$
$w^1 \equiv 1$, w^2 induces A decreasing proc. time order	reduces to a special MAX m-CUT problem
$w^1 \equiv 1$, w^2 arbitrary m constant	FPTAS of Sahni for $Pm\|\|\sum_j w_j C_j$ can be generalized

The structure of optimal solutions

Lemma

There is a global ordering of jobs such that in an optimal solution on each machine the job sequence respects the global order.
In the optimistic case the global order is WSPT with respect to w^2 and in case of ties WSPT w.r.t. w^1.
In the pessimistic case the global order is WSPT with respect to w^2 and in case of ties reverse WSPT w.r.t. w^1.

Reduction to the MAX m-CUT problem

MAX m-CUT (optimization version)
input: the number of vertices (of a complete graph) n, edge weights c_e for all the $n(n-1)/2$ edges, a number m with $m \leq n$ (all data in $\mathbb{Z}_+$)
output: a partitioning of the vertices into m disjoint classes $V_1, \ldots, V_m$ such that the total weight of edges between the classes is maximized, i.e.,

$$\max_{(V_1,\ldots,V_m)} \sum_{k=1}^{m-1} \sum_{\ell=k+1}^{m} \sum_{i \in V_k, j \in V_\ell} c_{ij}$$

where the maximum is over all m-partitions of the n nodes
Reduction: the nodes are identified with the n tasks, and

$$c_{jk} = p_j w_k^1 \quad \text{if } \frac{w_j^2}{p_j} > \frac{w_k^2}{p_k}; \text{ or } \frac{w_j^2}{p_j} = \frac{w_k^2}{p_k} \text{ and } \frac{w_j^1}{p_j} \geq \frac{w_k^1}{p_k}$$

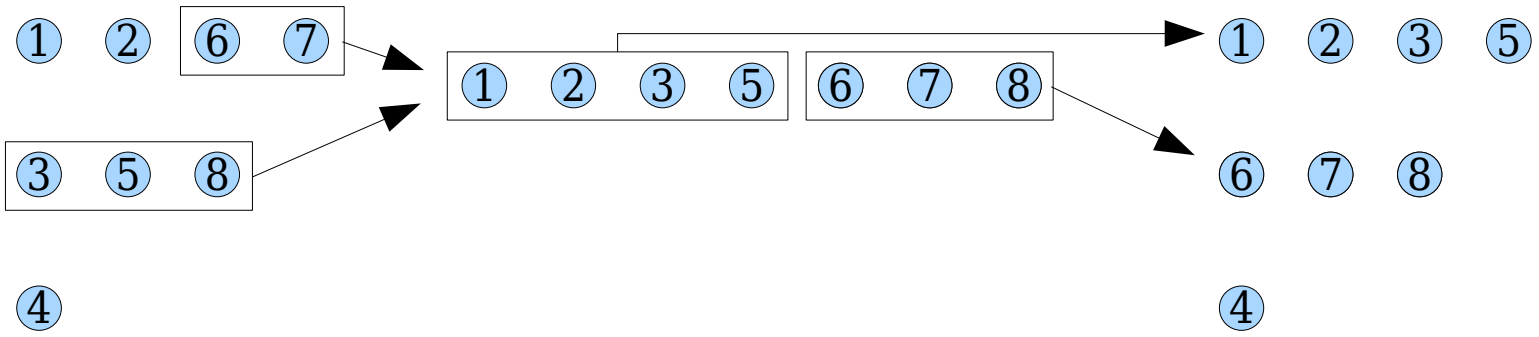
A special MAX m-CUT problem

Special weights

If $w^1 \equiv 1$ and $\frac{w_j^2}{p_j} > \frac{w_k^2}{p_k}$ iff $p_j > p_k$, then $c_{jk} = \max\{p_j, p_k\}$

Theorem

There exists an optimal solution to MAX m-CUT such that $V_1 = \{1, \ldots, k_1\}, V_2 = \{k_1 + 1, \ldots, k_2\}, \ldots, V_m = \{k_{m-1} + 1, \ldots, m\}$, *where* $p_j \geq p_k$ *for* $j < k$.



Corollary

The MAX m-CUT problem with the above weights can be solved by dynamic programming in polynomial time

The bilevel order acceptance problem

- There are n jobs with processing times p_j, due-dates d_j, and job-weights w_j^1, w_j^2; and a single machine
- Leader: selects a subset of jobs A (accepted jobs) to maximize $\sum_{j \in A} w_j^1$
- Follower: sequences the jobs non-preemptively to minimize $\sum_{j \in A} w_j^2 C_j$
- The solution is feasible iff the optimal solution chosen by the Follower meets the due-dates of all jobs in A
- If the Leader is optimistic, it selects A such that at least one optimal solution of the Follower meets all the due-dates
- If the Leader is pessimistic, it selects A such that all the optimal solutions of the Follower with respect to A meets all the due-dates

Results on the bilevel order acceptance problem

Restriction	**Complexity**
no restriction	decision version is NP-complete solvable in pseudo-poly time
$w^1 \equiv 1$	Polynomial (generalized Moore-Hodgson alg.)

A polynomial algorithm for the $w^1 \equiv 1$ case

The Moore-Hodgson algorithm for $1||\sum U_j$

1. Order the jobs in EDD order: $d_1 \leq \ldots \leq d_n$
2. Starting with the first job, process the jobs one-by-one. If all jobs can be completed on time, stop. Otherwise, let k_1 be the first job such that $\sum_{j=1}^{k_1} p_j > d_{k_1}$. Remove from the first k_1 jobs the one with largest p_j value, and proceed with the next job.

Modification for the bilevel order acceptance problem:

1. Order the jobs in the Follower's WSPT order: $j < k$ iff $\frac{w_j^2}{p_j} > \frac{w_k^2}{p_k}$ and in case of ties if $d_j < d_k$ $(d_j > d_k)$

Publications

- Kis, T., & Kovcs, A. (2012). On bilevel machine scheduling problems. OR spectrum, 34(1), 43-68.
- Kovcs, A., & Kis, T. (2011). Constraint programming approach to a bilevel scheduling problem. Constraints, 16(3), 317-340.
- Kis, T., & Kovcs, A. (2013). Exact solution approaches for bilevel lot-sizing. European Journal of Operational Research, 226(2), 237-245.