A Serial Schedule-Generation Scheme for Preemptive Project Scheduling Problems with Generalized Precedence Relations and Regular Min-Max Criteria

Christoph Schwindt
Tobias Paetz

Operations Management Group
Clausthal University of Technology

15th International PMS Conference, Valencia

TU Clausthal
Clausthal University of Technology

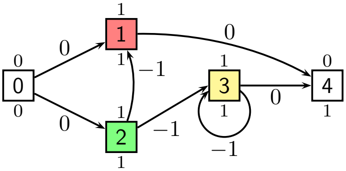
PMS 2016

Outline

Preemptive project scheduling problem

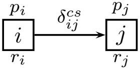
- Project consists of preemptive activities $i \in V$ with durations p_i
- Activities i require r_{ik} units of renewable resources $k \in \mathcal{R}$ with capacities R_k
- Minimum time lags δ_{ij}^{cs} between completion time C_i of activity i and start time S_j of activity j with $(i, j) \in E \subseteq V \times V$
  - $-\delta_{ij}^{cs} > 0$: maximum start-to-completion time lag between j and i
  - $-\delta_{ii}^{cs} \geq p_i$: maximum makespan of i
- Sought: feasible schedule $x : t \mapsto (x_j(t))_{j \in V}$ minimizing objective function $f_{max}(C) = \max_{j \in V} f_j(C_j)$ with regular f_j
  - $x_j(t)$: percentage of activity j processed by time $t \geq 0$
  - $S_j = \sup\{t \geq 0 \mid x_j(t) = 0\}$, $C_j = \inf\{t \geq 0 \mid x_j(t) = 1\}$
  - $y_j(t) := p_j \cdot \frac{\mathrm{d}^+ x_j}{\mathrm{d}t}(t) = \begin{cases} 1, & \text{if } j \text{ is in progress at time } t \\ 0, & \text{otherwise} \end{cases}$

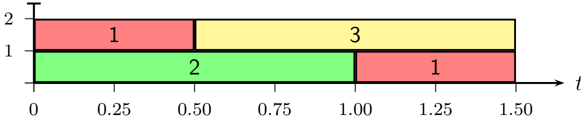
Example

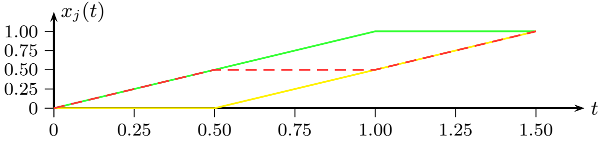


$\boxed{R = 2,\ f_{max} = C_{max}}$

Legend:







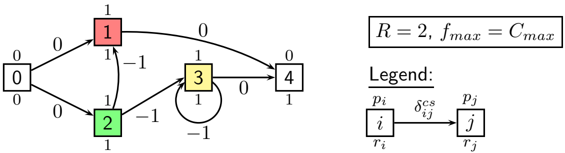
Conceptual model

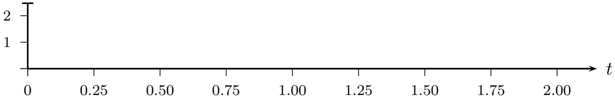
$$(P)\left\{\begin{array}{lll} \text{Minimize} & f_{max}(C) = \max_{j\in V} f_j(C_j) & \\ \text{subject to} & r_k(t) := \sum_{i\in V} r_{ik} y_i(t) \leq R_k & (k \in \mathcal{R};\ t \geq 0) \\ & S_j \geq C_i + \delta_{ij}^{cs} & ((i,j) \in E) \\ & S_j \geq 0,\ C_j \geq S_j + p_j & (j \in V) \end{array}\right.$$

- Non-preemptive problem contained as a special case ($\delta_{ii}^{cs} = -p_i$)
- Feasibility variant strongly NP-hard
- By convention $\delta_{ij}^{cs} \leq 0$ ($\delta_{ij}^{cs} > 0$ as dummy h with $p_h = \delta_{ij}^{cs}$)
- Each activity interrupted at most $n-1$ times
- Schedule encoded as set Σ of at most n^2 triples (j, s_j, c_j) defining time intervals $[s_j, c_j[$ during which parts of activities j are in progress

Why naïve adaption of SGS by Franck et al. (2001) fails

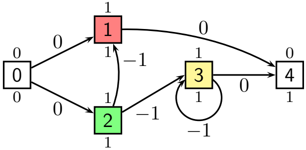
- In each iteration select eligible activity j^* to be started or resumed
- Schedule j^* at earliest time- and resource-feasible point in time
- Allow to suspend execution of j^* at next decision point (release, start, or completion of some activity)





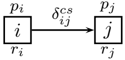
Why naïve adaption of SGS by Franck et al. (2001) fails

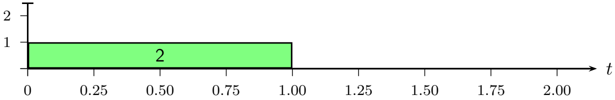
- In each iteration select eligible activity j^* to be started or resumed
- Schedule j^* at earliest time- and resource-feasible point in time
- Allow to suspend execution of j^* at next decision point (release, start, or completion of some activity)



$R = 2,\ f_{max} = C_{max}$

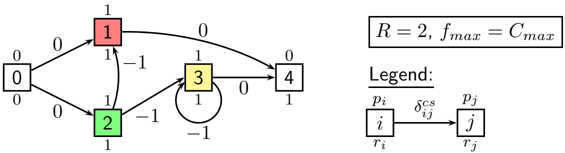
Legend:

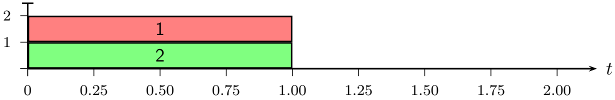




Why naïve adaption of SGS by Franck et al. (2001) fails

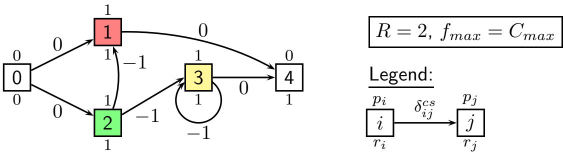
- In each iteration select eligible activity j^* to be started or resumed
- Schedule j^* at earliest time- and resource-feasible point in time
- Allow to suspend execution of j^* at next decision point (release, start, or completion of some activity)

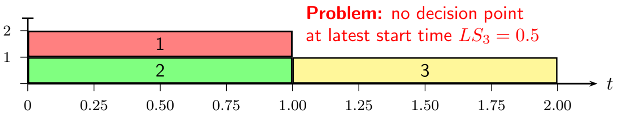




Why naïve adaption of SGS by Franck et al. (2001) fails

- In each iteration select eligible activity j^* to be started or resumed
- Schedule j^* at earliest time- and resource-feasible point in time
- Allow to suspend execution of j^* at next decision point (release, start, or completion of some activity)





- Need additional decision points arising from objective function value

The solution: iterate threshold instances

- Given upper bound v on objective function value $f_{max}(C)$,

$$f_{max}(C) = \max_{j \in V} f_j(C_j) \leq v \Leftrightarrow f_j(C_j) \leq v \ (j \in V)$$
$$\Leftrightarrow C_j \leq f_j^{-1}(v) \ (j \in V)$$

with $f_j^{-1}(v) := \sup\{C_j \mid f_j(C_j) \leq v\}$ and $\sup \emptyset := -\infty$

- Introducing $-\delta_{j0}^{cs} = f_j^{-1}(v)$ for $j \in V$ ensures $f_{max}(C) \leq v$
- Perform binary search for smallest v over interval $[lb, ub]$
  - Start with $v = \frac{lb+ub}{2}$
  - Apply preemptive SGS to instance with $-\delta_{j0}^{cs} = f_j^{-1}(v)$
    - If feasible schedule is found: $f_{max}^* \leq f_{max}(C) \leq v$, put $ub := f_{max}(C)$
    - Otherwise: put $lb := v$
  - Recursively continue search on $[lb, ub]$ until $ub - lb \leq \varepsilon$

Basic principle of preemptive SGS

- Translate upper bound v into time lags $-\delta_{j0}^{cs} = f_j^{-1}(v)$ for $j \in V$
- Compute distance matrix $D^{cs} = (d_{ij}^{cs})_{i,j\in V}$ of transitive time lags
  - $ES_j = d_{0j}^{cs}$: earliest time-feasible start time of activity j
  - $LC_j = -d_{j0}^{cs}$: latest time-feasible completion times of activity j
- Initialize earliest time-feasible start times $es_j := ES_j$ of pending parts of activities j
- Activity j eligible for being started or resumed if all predecessors $i \in Pred^{\prec}(j)$ have been completed, i. e., $Pred^{\prec}(j) \subseteq \mathcal{C}$, where

$i \prec j \Leftrightarrow i$ must be started no later than j but not vice versa

$$\Leftrightarrow \max\{d_{ij}^{cs}, d_{i0}^{cs} + es_j\} + p_i \geq 0 \wedge \max\{d_{ji}^{cs}, d_{j0}^{cs} + es_i\} + p_j < 0$$

- Select some eligible activity $j^* \in \mathcal{E}$ by applying priority rule π

Basic principle of preemptive SGS

- Determine earliest resource-feasible start time $s_{j^*} \geq es_{j^*}$
- If $s_{j^*} \leq LC_{j^*} - p_{j^*}$, execute j^* up to next decision point c_{j^*}
- Set $\mathcal{D}$ of decision points contains
  - earliest completion time $s_{j^*} + p_{j^*}$ of j^*
  - start and completion times s_j, c_j of executed activity parts
  - earliest and latest start and completion times $es_j, es_j + p_j, LC_j - p_j, LC_j$ of pending activity parts
- Decrease residual processing time p_{j^*} by $c_{j^*} - s_{j^*}$ and update earliest start and latest completion times of pending activities
- If $s_{j^*} > LC_{j^*} - p_{j^*}$, try to repair the schedule by calling unscheduling procedure
- Proceed until all activities $j \in V$ have been entirely added to the schedule, i. e., $\mathcal{C} = V$

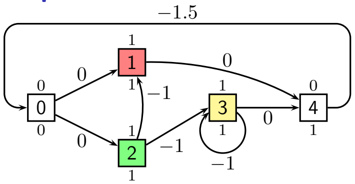
Basic principle of unscheduling procedure

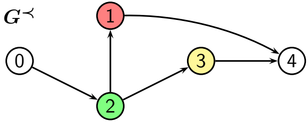
- If $s_{j^*} > LC_{j^*} - p_{j^*}$, SGS has run into a deadlock
- Clear parts of the schedule to remove deadlock
- Identify set $\mathcal{U}$ of started activities $j \in \mathcal{S}$ that determined the latest completion time LC_{j^*} of activity j^*

$$\mathcal{U} = \{j \in \mathcal{S} \mid LC_{j^*} = S_j - d^{cs}_{j^*j}\}$$

- Remove all activity parts with $s_j \geq \min_{i \in \mathcal{U}} S_i$ from the schedule
- Delay activities $j \in \mathcal{U}$ by $\Delta = s_{j^*} + p_{j^*} - LC_{j^*}$: put $es_j := S_j + \Delta$
- If $es_j + p_j > -d^{cs}_{j0}$, deadlock could not be resolved: terminate
- Otherwise return to preemptive SGS

Example

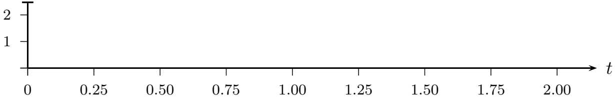




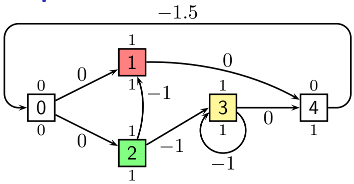
$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

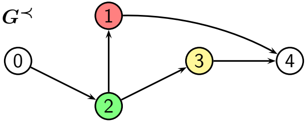
j	1	2	3
es_j	0	0	0
LC_j	1.5	1.5	1.5
S_j	–	–	–
C_j	–	–	–

$\mathcal{S} = \{0\}$
$\mathcal{C} = \{0\}$



Example

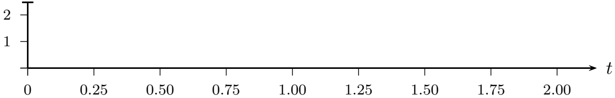




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

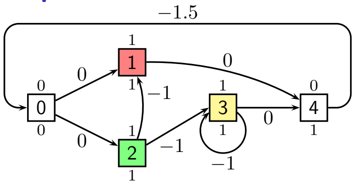
j	1	2	3
es_j	0	0	0
LC_j	1.5	1.5	1.5
S_j	–	–	–
C_j	–	–	–

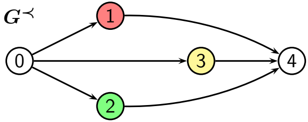
$\mathcal{S} = \{0\}$
$\mathcal{C} = \{0\}$



$$\mathcal{E} = \{2\},\ j^* = 2,\ s_2 = 0,\ \mathcal{D} = \{0, 0.5, 1, 1.5\},\ c_2 = 0.5,\ p_2 = 0.5$$

Example

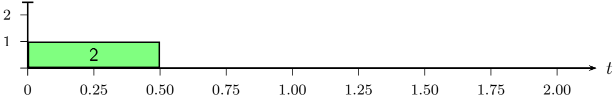




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

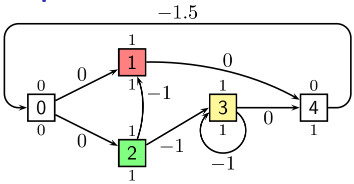
j	1	2	3
es_j	0	0.5	0
LC_j	1.5	1.5	1.5
S_j	–	0	–
C_j	–	–	–

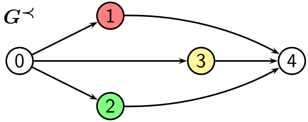
$\mathcal{S} = \{0, 2\}$
$\mathcal{C} = \{0\}$



$\mathcal{E} = \{1, 2, 3\}$, $j^* = 3$, $s_3 = 0$, $\mathcal{D} = \{0, 0.5, 1, 1.5\}$, $c_3 = 0.5$, $p_3 = 0.5$

Example

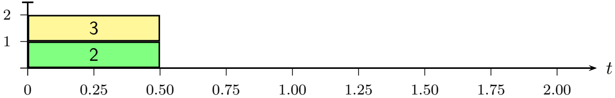




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

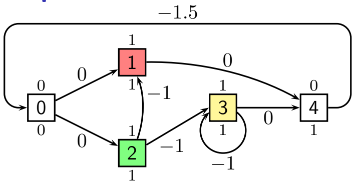
j	1	2	3
es_j	0	0.5	0.5
LC_j	1.5	1	1
S_j	–	0	0
C_j	–	–	–

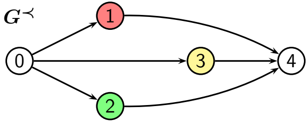
$\mathcal{S} = \{0, 2, 3\}$
$\mathcal{C} = \{0\}$



$\mathcal{E} = \{1, 2, 3\}$, $j^* = 1$, $s_1 = 0.5$, $\mathcal{D} = \{0, 0.5, 1, 1.5\}$, $c_1 = 1$, $p_1 = 0.5$

Example

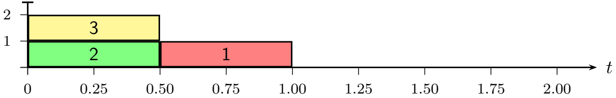




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

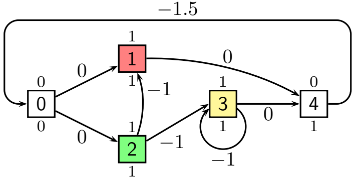
j	1	2	3
es_j	1	0.5	0.5
LC_j	1.5	1	1
S_j	0.5	0	0
C_j	–	–	–

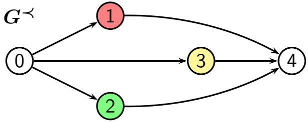
$\mathcal{S} = \{0, 1, 2, 3\}$
$\mathcal{C} = \{0\}$



$$\mathcal{E} = \{1, 2, 3\},\ j^* = 3,\ s_3 = 0.5,\ \mathcal{D} = \{0, 0.5, 1, 1.5\},\ c_3 = 1,\ p_3 = 0$$

Example

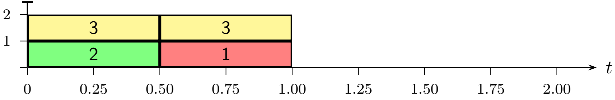




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

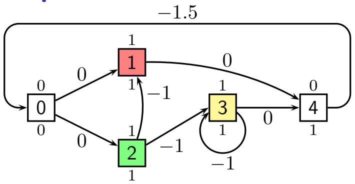
j	1	2	3
es_j	1	0.5	0.5
LC_j	1.5	1	1
S_j	0.5	0	0
C_j	–	–	1

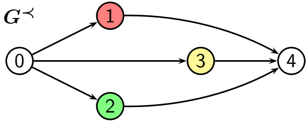
$\mathcal{S} = \{0, 1, 2, 3\}$
$\mathcal{C} = \{0, 3\}$



$\mathcal{E} = \{1, 2\},\ j^* = 2,\ s_2 = 1,\ \mathcal{D} = \{0, 0.5, 1, 1.5\}$

Example

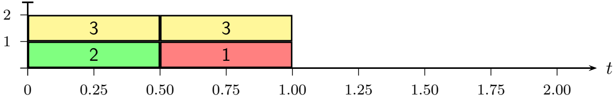




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

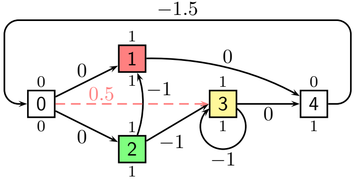
j	1	2	3
es_j	1	0.5	0.5
LC_j	1.5	1	1
S_j	0.5	0	0
C_j	–	–	1

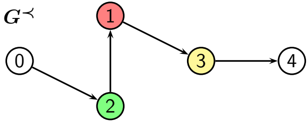
$\mathcal{S} = \{0,1,2,3\}$
$\mathcal{C} = \{0,3\}$



$s_2 + p_2 > LC_2$: *unschedule*$(2, 0.5)$, $\mathcal{U} = \{3\}$, $t^* = S_3 = 0$, $es_3 = 0.5$

Example

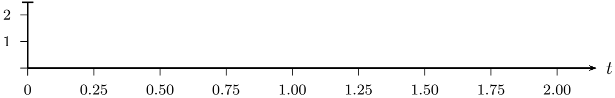




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

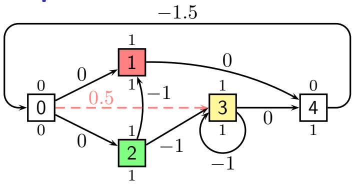
j	1	2	3
es_j	0	0	0.5
LC_j	1.5	1.5	1.5
S_j	–	–	–
C_j	–	–	–

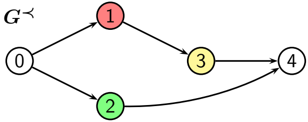
$\mathcal{S} = \{0\}$
$\mathcal{C} = \{0\}$



$\mathcal{E} = \{2\},\ j^* = 2,\ s_2 = 0,\ \mathcal{D} = \{0, 0.5, 1, 1.5\},\ c_2 = 0.5,\ p_2 = 0.5$

Example

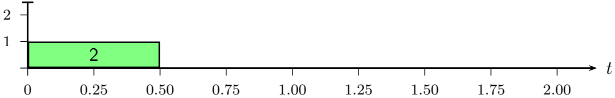




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

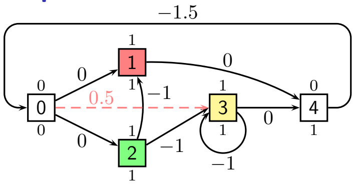
j	1	2	3
es_j	0	0.5	0.5
LC_j	1.5	1.5	1.5
S_j	–	0	–
C_j	–	–	–

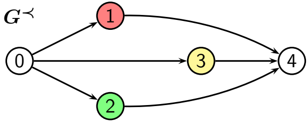
$\mathcal{S} = \{0, 2\}$
$\mathcal{C} = \{0\}$



$\mathcal{E} = \{1, 2\},\ j^* = 1,\ s_1 = 0,\ \mathcal{D} = \{0, 0.5, 1, 1.5\},\ c_1 = 0.5,\ p_1 = 0.5$

Example

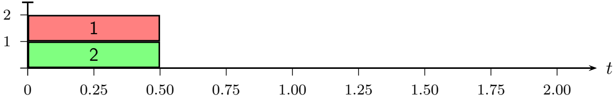




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

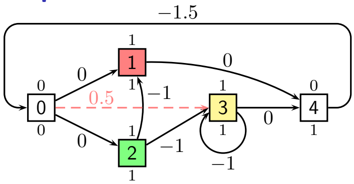
j	1	2	3
es_j	0.5	0.5	0.5
LC_j	1.5	1	1.5
S_j	0	0	–
C_j	–	–	–

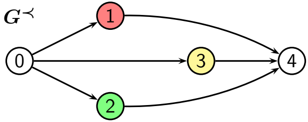
$\mathcal{S} = \{0, 1, 2\}$
$\mathcal{C} = \{0\}$



$\mathcal{E} = \{1, 2, 3\},\ j^* = 2,\ s_2 = 0.5,\ \mathcal{D} = \{0, 0.5, 1, 1.5\},\ c_2 = 1,\ p_2 = 0$

Example

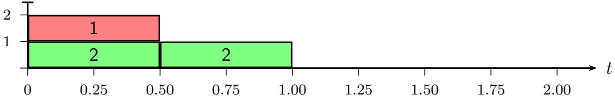




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

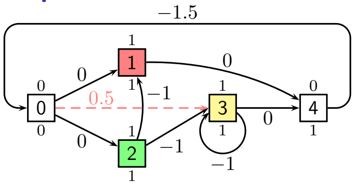
j	1	2	3
es_j	0.5	0.5	0.5
LC_j	1.5	1	1.5
S_j	0	0	–
C_j	–	1	–

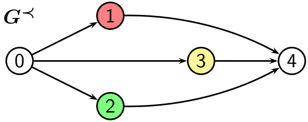
$\mathcal{S} = \{0, 1, 2\}$
$\mathcal{C} = \{0, 2\}$



$\mathcal{E} = \{1, 3\},\ j^* = 3,\ s_3 = 0.5,\ \mathcal{D} = \{0, 0.5, 1, 1.5\},\ c_3 = 1,\ p_3 = 0.5$

Example

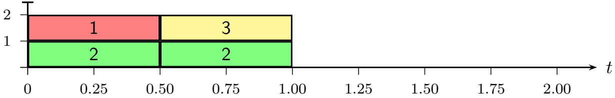




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

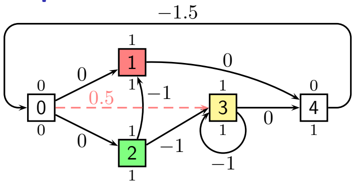
j	1	2	3
es_j	0.5	0.5	1
LC_j	1.5	1	1.5
S_j	0	0	0.5
C_j	–	1	–

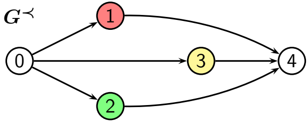
$\mathcal{S} = \{0, 1, 2, 3\}$
$\mathcal{C} = \{0, 2\}$



$\mathcal{E} = \{1, 3\},\ j^* = 1,\ s_1 = 1,\ \mathcal{D} = \{0, 0.5, 1, 1.5\},\ c_1 = 1.5,\ p_1 = 0$

Example

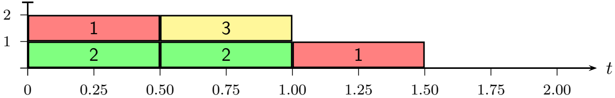




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

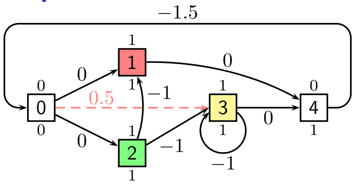
j	1	2	3
es_j	0.5	0.5	1
LC_j	1.5	1	1.5
S_j	0	0	0.5
C_j	1.5	1	–

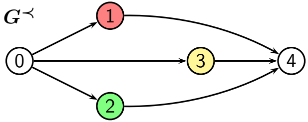
$\mathcal{S} = \{0, 1, 2, 3\}$
$\mathcal{C} = \{0, 1, 2\}$



$\mathcal{E} = \{3\},\ j^* = 3,\ s_3 = 1,\ \mathcal{D} = \{0, 0.5, 1, 1.5\},\ c_3 = 1.5,\ p_3 = 0$

Example

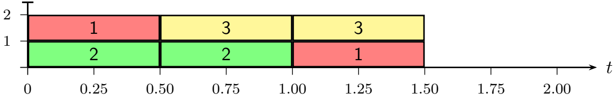




$$D^{cs} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \\ -1.5 & -1 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1 & 0 \\ -1.5 & -1.5 & -1.5 & -1.5 & 0 \end{pmatrix}$$

j	1	2	3
es_j	0.5	0.5	1
LC_j	1.5	1	1.5
S_j	0	0	0.5
C_j	1.5	1	1.5

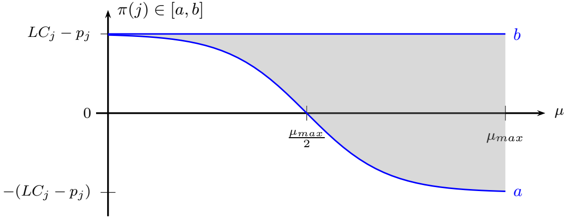
$\mathcal{S} = \{0, 1, 2, 3\}$
$\mathcal{C} = \{0, 1, 2, 3\}$



Randomized multi-start procedure

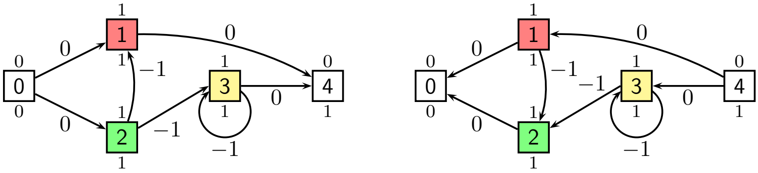
- In multi-start procedure, binary search algorithm can be run with different priority rules π
- Start with minimum latest start time rule: $\pi(j) = LC_j - p_j$
- Stepwise increase random bias according to sigmoid function $\sigma(\mu)$ for passes $\mu = 1, \ldots, \mu_{max}$

$$\pi(j) = (LC_j - p_j) \cdot (1 - 2\sigma(\mu) \cdot rnd) \text{ with } \sigma(\mu) = \frac{1}{1+e^{-\alpha\cdot(\mu-\mu_{max}/2)}}$$



Forward and backward scheduling

- Scheduling pending activity parts at their earliest feasible start times corresponds to forward scheduling
- Backward scheduling can be emulated by applying SGS on "inverted" project network (Hanzálek and Šůcha 2009)
- Substitute time lags δ_{ij}^{cs} into time lags $\bar{\delta}_{ji}^{cs} = \delta_{ij}^{cs}$
- Original and inverted instances $I, \bar{I}$ of threshold problem equivalent
- Schedule $\bar{\Sigma}$ for $\bar{I}$ transformed into schedule Σ for I by replacing triples $(j, \bar{s}_j, \bar{c}_j) \in \bar{\Sigma}$ with triples $(j, \bar{C}_{max} - \bar{c}_j, \bar{C}_{max} - \bar{s}_j)$



Experimental performance analysis

- Testsets UBO10, UBO20, UBO50, UB100 with 90 instances each (Franck et al. 2001)
- Objective function project duration $f_{max}(C) = C_{max}$
- Lower and upper bounds for preemptive instances computed with MILP formulation (S. and Paetz 2015)
- Upper bounds for non-preemptive instances according to benchmark files on ProGen/max homepage[1]
- Tested configurations
  - Forward scheduling: $\mu_{max} = 100$, $\varepsilon = 10^{-4}$
  - Backward and forward scheduling: $\mu_{max} = 50$, $\varepsilon = 10^{-4}$
- Multi-start procedure coded in C#
- Intel i5 PC with 3.4 GHz clock pulse and 8 GB RAM, OS Win 7 Professional 64 Bit

[1] www.wiwi.tu-clausthal.de/abteilungen/produktion/forschung/schwerpunkte/project-generator/

Computational results

Forward scheduling, 100 passes

	Δ_{milp}	Δ_{npmtn}	p_{imp}	p_{fnd}	t_{cpu}	$\# it$
$n = 10$	0.50 %	−1.71 %	27.78 %	4.44 %	0.14 s	1668
$n = 20$	0.06 %	−1.68 %	36.67 %	10.00 %	0.37 s	1746
$n = 50$	n/a	−1.35 %	44.44 %	3.33 %	3.15 s	1728
$n = 100$	n/a	−1.16 %	46.67 %	0.00 %	24.50 s	1837

Forward and backward scheduling, 50 passes

	Δ_{milp}	Δ_{npmtn}	p_{imp}	p_{fnd}	t_{cpu}	$\# it$
$n = 10$	0.17 %	−2.04 %	31.11 %	4.44 %	0.14 s	1662
$n = 20$	−1.08 %	−2.66 %	46.67 %	11.11 %	0.34 s	1715
$n = 50$	n/a	−2.12 %	52.22 %	2.22 %	3.37 s	1732
$n = 100$	n/a	−1.41 %	57.78 %	1.11 %	25.16 s	1835

Summary

- SGS for preemptive project scheduling problems with generalized precedence relations and regular min-max criteria
- Create new decision points by iterating threshold instances
- Multi-start heuristic with randomly biased priority indices
- Improvements on benchmark results for non-preemptive problems
  - Preemption gains obtained for 47 % of the instances
  - Feasible schedules generated for 17 out of 66 instances that are infeasible when preemption is not allowed

Future research

- Performance evaluation for other regular min-max criteria
- Metaheuristics embedding preemptive SGS

References

Franck B, Neumann K, Schwindt C (2001)
Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling
OR Spektrum 23: 297–324

Hanzálek Z, Šůcha P (2009)
Time Symmetry of Project Scheduling with Time Windows and Take-Give Resources
In: 4th Multidisciplinary International Scheduling Conference: Theory and Applications, pp 239–253

Schwindt C, Paetz T (2015)
Continuous Preemption Problems
In: Schwindt C, Zimmermann J (eds) Handbook on Project Management and Scheduling Vol. 1. Springer, Cham, Heidelberg, New York, Dordrecht, London, pp 251–295