§ 21 Generalized Project Scheduling and IP - Models

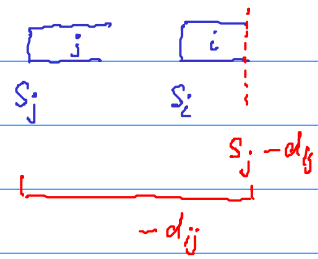
A. Generalizing $\underbrace{\text{precedence constraints}}$ by $\underbrace{\text{time lags}}$

$$i<j \iff \underset{\substack{\uparrow \\ \text{positive}}}{S_i + x_i} \leq S_j \qquad\qquad S_i + \underset{\substack{\uparrow \\ \text{may also be negative}}}{d_{ij}} \leq S_j$$

d_{ij} is called a time lag between jobs i and j

Interpretation of negative d_{ij}:

$$S_i + d_{ij} \leq S_j \iff S_i \leq S_j + \underset{\geq 0}{(-d_{ij})}$$



So S_j defines a deadline for the start of i

So time lags can express lower and upper bounds on the start times and completion times relative to other jobs

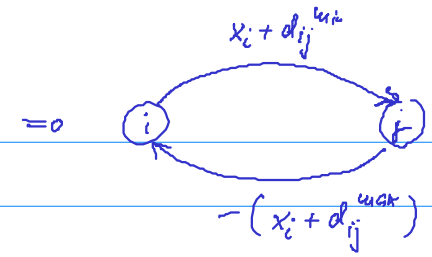
↳ because of $S_i + x_i = C_i$

Therefore, time lags are often referred to as minimum time lags (lower bounds) or maximum time lags (upper bounds)

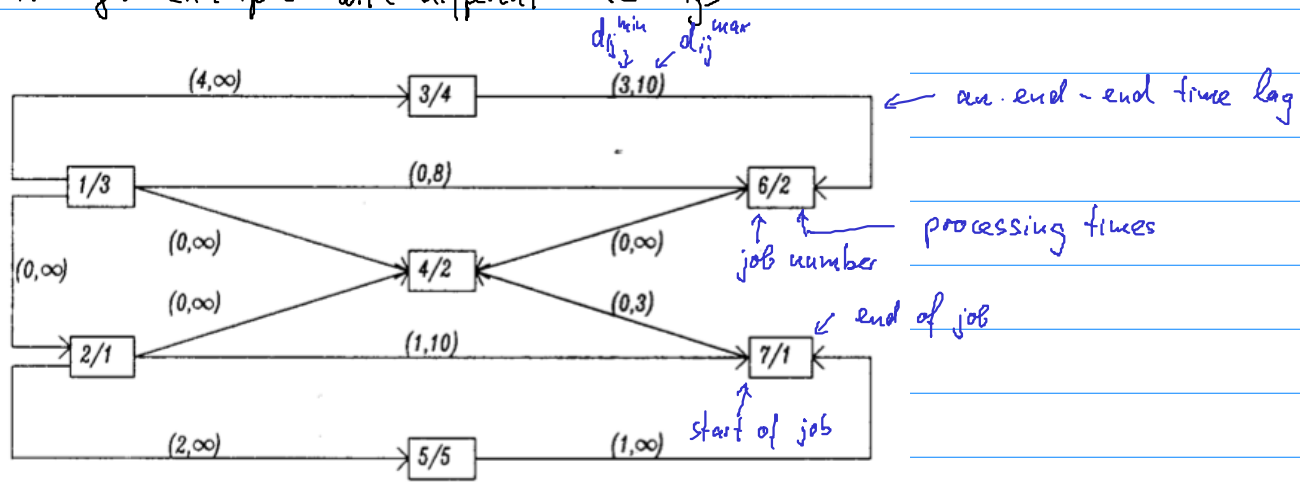
We will represent them in the standardized form $S_i + d_{ij} \leq S_j$ and represent this in a digraph by the arc $i \xrightarrow[d_{ij}]{} j$ with arc weight d_{ij}

Expl. $C_i + d_{ij}^{min} \leq S_j \leq C_i + d_{ij}^{max}$ $\quad (0 \leq d_{ij}^{min} \leq d_{ij}^{max})$

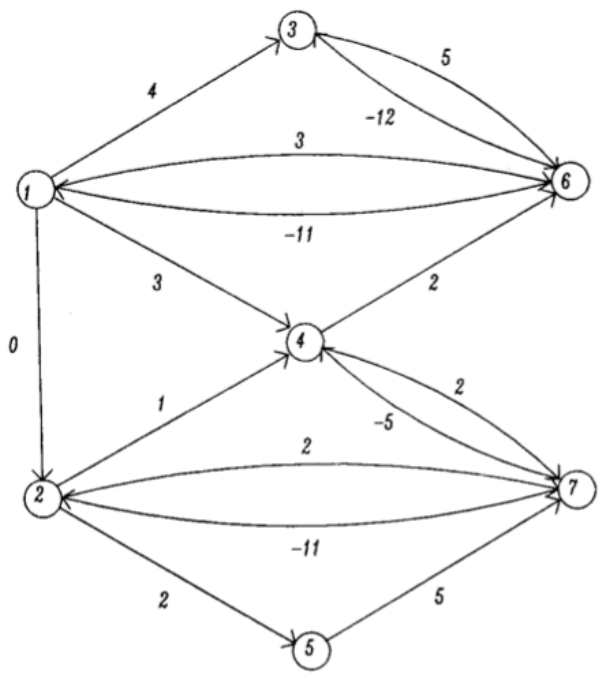
$S_i + x_i + d_{ij}^{min} \leq S_j \leq S_i + x_i + d_{ij}^{max}$



A larger example with different time lags



The resulting digraph



21.1 THEOREM

a) A project network with time lags has a feasible schedule
$\iff$ the associated digraph has no directed cycle of positive length

b) In that case, the earliest start of job j is the length of a longest path from an artificial start node s (with 0-arcs to all other nodes)

arcs with weight 0

to j

c) Checking for the existence of a positive cycle and computing the earliest start can be done in $O(n^3)$ time with the Moore – Bellman – Ford algorithm

Proof: see ADM I: existence of negative cycles, feasible potentials and conservative arc weights

B. Project scheduling with time lags and start-time dependent cost

Assume $V = \{0, 1, \dots, n\}$

project start project end both with processing time 0

$p_j > 0$ = fixed deterministic processing time of job j $p_j \in \mathbb{N}$

$L \subseteq V \times V$ set of start-to-start time lags $S_i + d_{ij} \leq S_j$

we assume that they admit a feasible schedule

w_{jt} = cost if job j is started at time t, $t = 0, 1, 2, \dots, T$

upper bound on makespan

objective: find a schedule that starts every job subject to the time lags and minimizes the total cost $\sum_j$ start time of job j

Formulation as IP

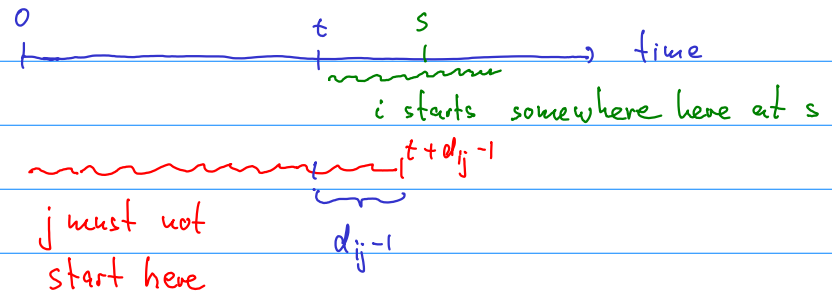
Variables $x_{jt} := \begin{cases} 1 & \text{if job } j \text{ starts at } t \\ 0 & \text{otherwise} \end{cases}$

Objective $\min \sum_j \sum_t w_{jt} x_{jt} =: w(x)$ (1)

such that $\sum_t x_{jt} = 1 \quad \forall j$ every job is started exactly once (2)

$$\sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+d_{ij}-1} x_{js} \leq 1 \quad \forall (i,j) \in L \quad \forall t \qquad (3)$$

every time lag is respected



so if i starts at S_i and j at S_j and $S_i + d_{ij} > S_j$
then (3) is violated for $t = S_i$

$$x_{jt} \in \{0, 1\} \qquad (4)$$

We compute for every job in advance
- earliest start times $e(j) \geq 0$
- latest start times $\ell(j) \leq T - p_j$

and set $x_{jt} = 0$ for $t \in [e(j), \ell(j)]$

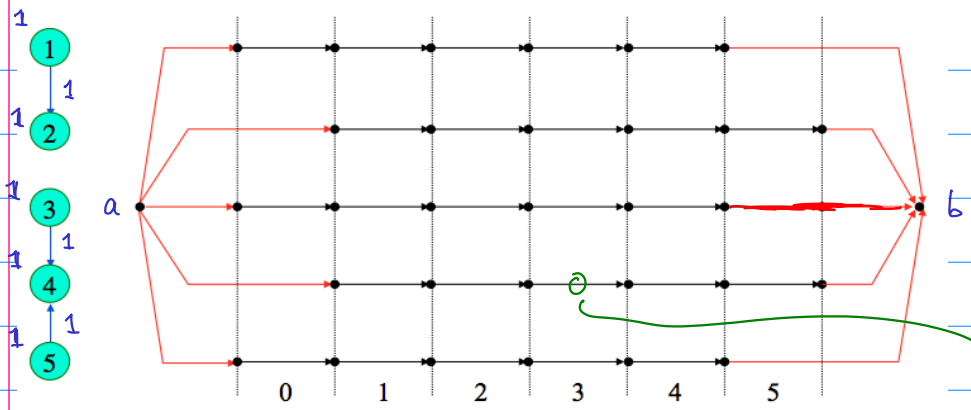
Because of (2), we lift all w_{jt} by a constant M such that
$\overline{w}_{jt} := w_{jt} + M \geq 0$
This changes the objective only by the additive constant $(n+1) \cdot M$
$\Rightarrow$ will assume w.l.o.g. that $w_{jt} \geq 0$

Transformation into a minimum cut problem

$D = (N, A)$ is the digraph for the min-cut problem
Nodes : one node v_{jt} $\forall$ jobs j and $\forall$ possible start times t and $t+1$
$N = \{ v_{jt} \mid j \in V \;\; t = e(j), \dots, \ell(j)+1 \} \cup \{a, b\}$ a = source b = sink

Arcs : A = assignment arcs + temporal arcs + auxiliary arcs
assignment arcs = $(v_{jt}, v_{j,t+1})$ $\forall j \in V$
auxiliary arcs = $(a, v_{j,e(j)})$ and $(v_{j,\ell(j)+1}, b)$ $\forall j$

all $p_j = 1$, all $d_{ij} = 1$



assignment arcs ⟶

auxiliary arcs ⟶

$x_{43} = 1$

i.e. $S_4 = 3$

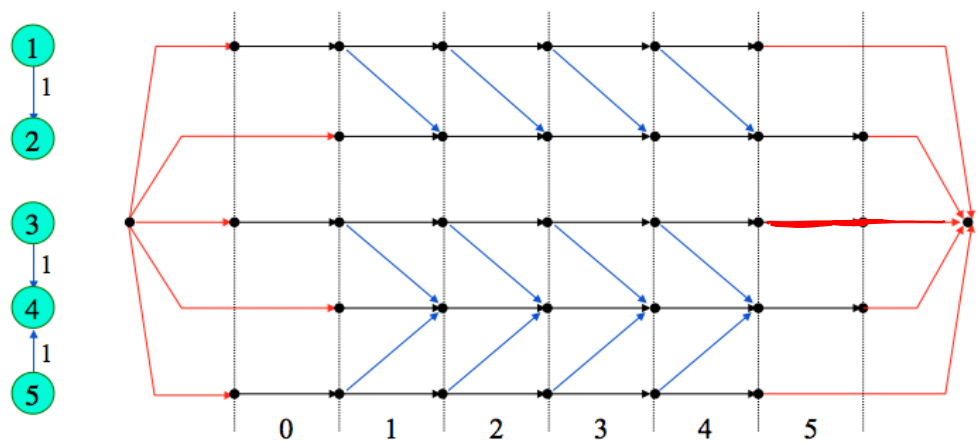
$T = 6$

assignment arcs represent start times

Temporal arcs $(v_{it}, v_{j,t+d_{ij}})$ for all time lags $(i,j) \in L$ and all t

with $e(i) + 1 \leq t \leq \ell(i)$

$e(j) + 1 \leq t + d_{ij} \leq \ell(j)$



temporal arcs ⟶

Arc capacities: assignment arcs $c(v_{jt}, v_{j,t+1}) := w_{jt} \geq 0$

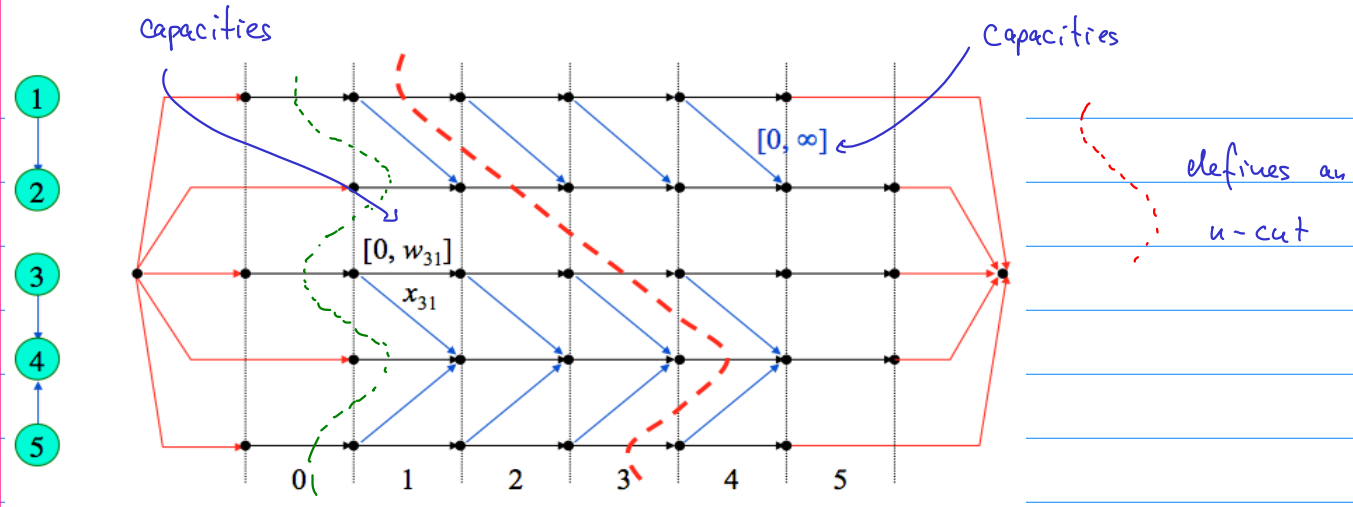
start cost of j at t

all other arcs: $c(\cdot) = \infty$

lower capacities of all arcs are 0

For our result, we are interested in special a,b-cuts:

An a,b-cut of D is an <u>n-cut</u> if the forward arcs of the cut contain exactly one assignment arc for every job



defines the n-cut with earliest starts
(exists by assumption)

21.2 Lemma (n-cuts are the important ones)
Let $(X, \bar{X})$ be a minimum a,b-cut of D and $c(X, \bar{X}) < \infty$
Then exists an n-cut $(X^*, \bar{X}^*)$ of D with the same value.
$(X^*, \bar{X}^*)$ can be computed in $O(n \cdot T)$ time from $(X, \bar{X})$

Proof

(1) $\boxed{(X, \bar{X}) \text{ contains at least one assignment arc for every job}}$
clear, otherwise $\exists$ directed path from a to b in X

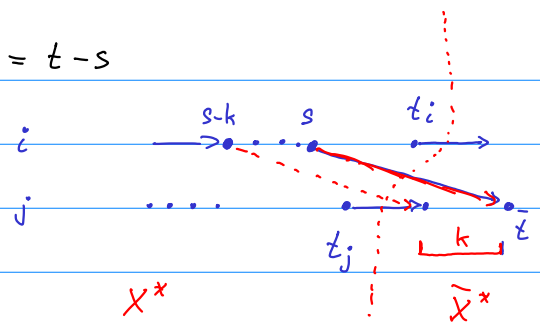
(2) $\boxed{\text{If } (X, \bar{X}) \text{ contains more than one assignment arc for some jobs, then we can construct from it an n-cut with the same capacity in } O(n \cdot T) \text{ time}}$

For such jobs j, let $t_j :=$ first index for which $v_{jt_j} \longrightarrow v_{j,t_j+1}$ is a forward arc in $(X, \bar{X})$
Define $X^* := \{a\} \cup \bigcup_j \{ v_{jt} \mid t \leq t_j \}$

(2.1) $\boxed{(X^*, \bar{X}^*) \text{ does not contain a temporal arc as forward arc}}$

Suppose it does contain $v_{is} \longrightarrow v_{jt}$ as forward arc $\hat{=}$ time lag $(i,j) \in L$

$d_{ij} = t - s$



$\Downarrow$

$s \leq t_i \;,\; t \geq t_j + 1$

Let $k := t - (t_j + 1) \geq 0$
t_j is possible start time for j $\Big\} \Rightarrow \boxed{e(j) + 1 \leq t - k}$ (i)

$\boxed{s - k \leq \ell(i)}$ (ii)

since $s - k \leq s \leq t_i \leq \ell(i)$

$\boxed{s - k \geq e(i) + 1}$ (iii)

otherwise $s - k < e(i) + 1$

$\Rightarrow t - k = s - k + (t - s) < e(i) + 1 + (t - s)$
(i,j) time lag with $d_{ij} = t - s \Rightarrow e(i) + (t - s) \leq e(j)$ $\Big\} \Rightarrow t - k < e(j) + 1$

contradiction to (i)

(iii) $\Rightarrow$ $v_{i,s-k}$ is a start node for i and $v_{i,s-k} \in X^*$
Definition of X^* $\Rightarrow$ $v_{i,s-k} \in X$
(i) $\Rightarrow$ $v_{j,t-k} = v_{j,t_j+1} \in \bar{X}$

since the arc $v_{j,t_j} \to v_{j,t_j+1}$ is forward arc in $(X, \bar{X})$

So the temporal arc $v_{i,s-k} \to v_{j,t_j+1}$ is a forward arc of $(X, \bar{X})$
$\Rightarrow c(X, \bar{X}) = \infty$, a contradiction $\Rightarrow$ (2.1)

$\boxed{\text{(2.2) } (X^*, \bar{X}^*) \text{ has the same capacity as } (X, \bar{X})}$

clear, since $(X^*, \bar{X}^*)$ contains fewer assignment arcs, no temporal arc,

and $w_{jt} \geq 0$

(3) $(X^*, \bar{X}^*)$ can be constructed in $O(n \cdot T)$ time from $(X, \bar{X})$ $\square$

21.3 THEOREM

There is a 1-1 correspondence between n-cuts $(X, \bar{X})$ with finite capacity and feasible solutions of the IP (1)–(4), namely

$$x_{jt} = \begin{cases} 1 & \text{if } (v_{jt}, v_{j,t+1}) \text{ is forward arc in } (X, \bar{X}) \\ 0 & \text{otherwise} \end{cases}$$

The capacity $c(X, \bar{X})$ is equal to the cost $w(x)$ of the IP

21.4 Corollary

The scheduling problem with start-dependent cost can be solved by computing a maximum a,b-flow and a corresponding a,b-cut in the digraph D

Proof Thm 21.3

(1) feasible solutions x of the IP yield cuts $(X, \bar{X})$ with $w(x) = c(X, \bar{X})$

x feasible $\Rightarrow$ exactly one $x_{j,t_j} = 1$ $\forall j$

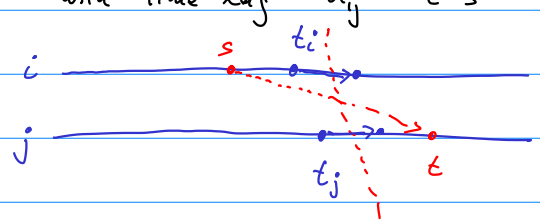
Set $X := \{a\} \cup \bigcup_j \{ v_{jt} \mid t \leq t_j \}$ $\longrightarrow$ n-cut

sum of capacities of assignment arcs gives $w(x)$

Suppose there is still another forward arc in $(X, \bar{X})$

$\Rightarrow$ this must be a temporal arc (v_{is}, v_{jt}) $s \leq t_i$, $t \geq t_j + 1$

with time lag $d_{ij} = t - s$



$\Rightarrow t_j - t_i < t - s = d_{ij}$, a contradiction to $t_i + t - s \leq t_j$

since x was a feasible solution

(2) every n-cut $(X, \bar{X})$ with finite capacity yields a feasible solution x of the IP

C Lower bounds for resource constrained project scheduling

We consider the RCPSP with time lags and C_{max} as objective
We can enhance the IP of subsection C :

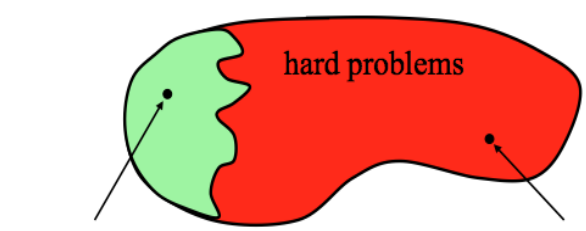
time-indexed binary variables $x_{jt} = \begin{cases} 1 : \text{job } j \text{ starts at time } t \\ 0 : \text{otherwise} \end{cases}$

$\min \sum_t t \cdot x_{n,t}$	minimize makespan	$\hat{=}$ (1) only job n has a cost value
$\sum_t x_{jt} = 1$	start every job j	$\hat{=}$ (2)
$\sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+d_{ij}-1} x_{js} \leq 1$	respect temporal distances d_{ij}	$\hat{=}$ (3)
$\sum_j r_{jk} \left(\sum_{s=t-p_j+1}^{t} x_{js} \right) \leq R_k$	respect resource constraints	$\forall k, \forall t$ ← new (5)
$x_{jt} \in \{0, 1\}$	0/1 variables	$\hat{=}$ (4)

[Pritzker, Watters, Wolfe 1969]

(5) says : sum of resource consumption of resource k at time t over all jobs j $\leq R_k$ = available amount of resource k

Simplify by relaxing the resource constraints



simplified problem ↳ will lead to B

original problem ← very hard, see § 20

We apply Lagrangian relaxation to the IP and relax inequalities (5)

$\min \sum_t t \cdot x_{n,t}$	minimize makespan
$\sum_t x_{jt} = 1$	start every job j
$\sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+d_{ij}-1} x_{js} \leq 1$	respect temporal distances d_{ij}
$\boxed{\sum_j r_{jk} \left(\sum_{s=t-p_j+1}^{t} x_{js} \right) \leq R_k}$	respect resource constraints
$x_{jt} \in \{0, 1\}$	0/1 variables

punish violation of capacity constraints in objective ← by $\lambda_{tk} \geq 0 \quad \forall k, t$

The objective becomes $\sum_t t x_{nt} + \sum_j \sum_t \left(\sum_{k \in R} r_{jk} \sum_{s=t}^{t+p_j-1} \lambda_{sk} \right) x_{jt} - \sum_t \sum_{k \in R} \lambda_{tk} R_k$

originally

$$\lambda_{tk} \left(\sum_j r_{jk} \left(\sum_{s=t}^{t+p_j-1} x_{js} \right) - R_k \right) \quad \forall k, t$$

We introduce weights

$$w_{jt} := \begin{cases} \sum_{k \in R} r_{jk} \sum_{s=t}^{t+p_j-1} \lambda_{sk} & \text{if } j \neq n, \\ w_{jt} := t & \text{if } j = n, \end{cases}$$

$\Rightarrow$ objective becomes $\sum_j \sum_t w_{jt} x_{jt} - \underbrace{\sum_t \sum_{k \in R} \lambda_{tk} R_k}_{\text{constant}}$

$\Rightarrow$ simplified problem is a scheduling problem with start-dependent cost

Its optimal value is a lower bound for the original problem

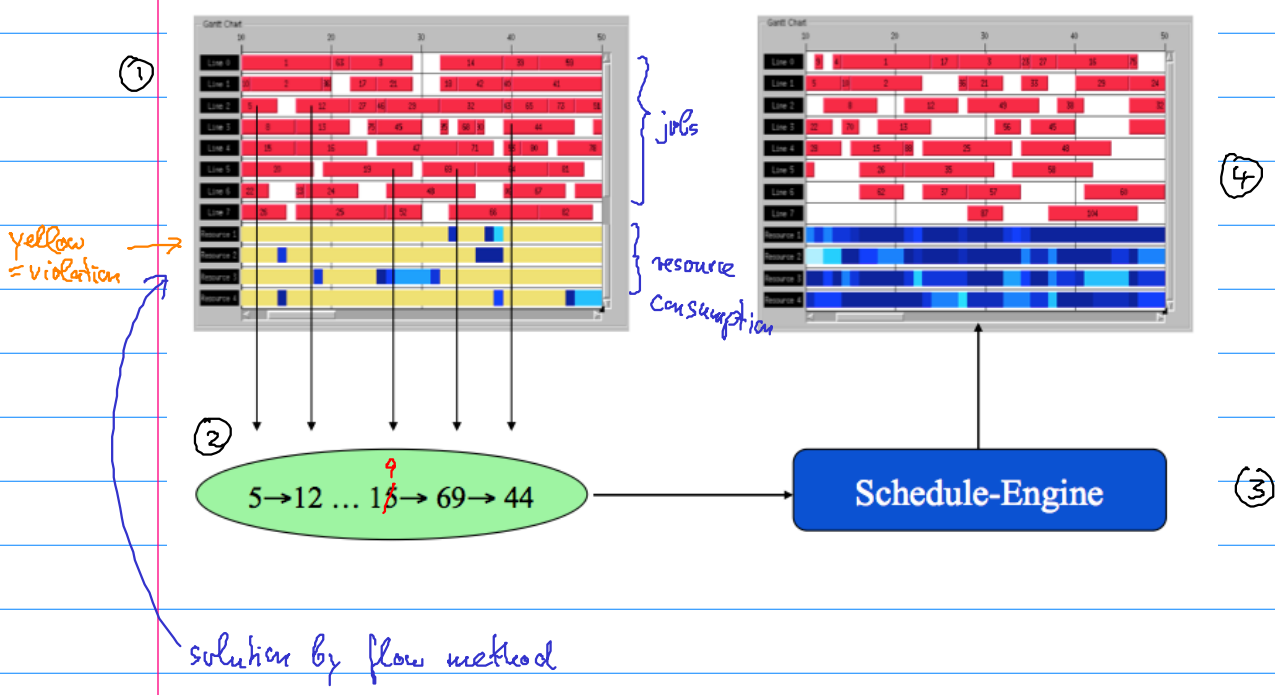
(since the optimal solution of the original problem is feasible for the modified problem with cost(modified problem) $\leq$ cost(original problem))

$\min \sum_{j,t} w_{jt} \cdot x_{jt}$	cost for makespan and resource violations
$\sum_{t} x_{jt} = 1$	start every job j
$\sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+d_{ij}-1} x_{js} \leq 1$	respect temporal distances d_{ij}
$x_{jt} \in \{0, 1\}$	0/1 variables

$\Rightarrow$ can be solved by max-flow computation

gives a time-feasible schedule, but not necessarily resource-feasible

From time-feasible to feasible schedules



1. time feasible schedule obtained from flow method (for fixed λ_{tk})
2. information extracted from time feasible schedule
   (precedences in the example)
3. generates from 2 a feasible schedule 4

The information (2) is a list $i_1, i_2, \ldots, i_n$ with $S_{i_1}(\alpha) \leq S_{i_2}(\alpha) \leq \ldots \leq S_{i_n}(\alpha)$

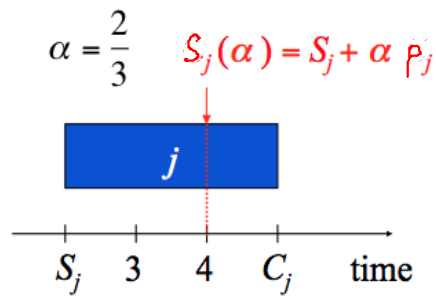
Schedule-Engine

Ideas from various areas of Operations Research

- serial list scheduling
- parallel list scheduling
- bidirectional scheduling
- improvement schemes
- scheduling by α-points
- …

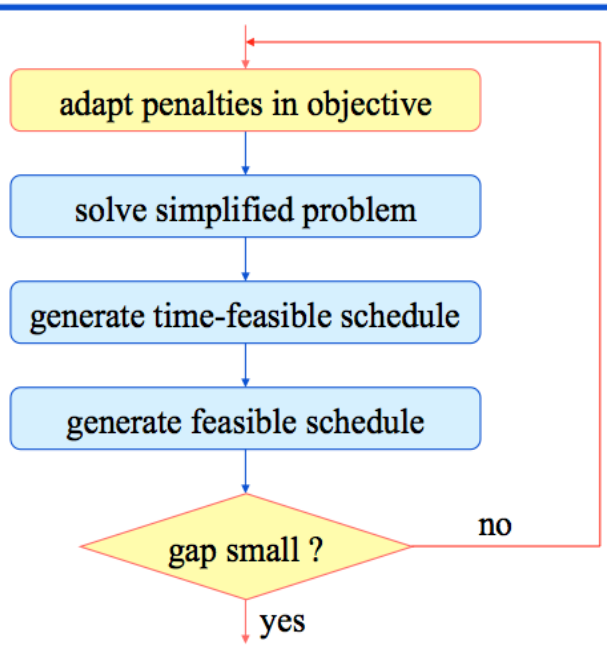
$0 \leq \alpha \leq 1$

$\alpha = \frac{2}{3}$



used by us for generating the list for several values of α
this gives different lists $L(\alpha)$
each such list is used for list scheduling (priority policy)

Lagrange-relaxation and subgradient optimization



← adapt the penalties λ_{tk}

Computational results: lower bounds

from 2001

600 instances with 120 jobs from PSPLIB

	CPU time	Above critical path
Our approach (≤ 220 iterations)	41 sec average 537 sec max	19.96 % average 146 % max
LP-relaxation	2241 sec average 23 h max	20.51 % average 155 % max
Best known	14.36 h average 72 h max	23.48 % average 168 % max

ES solution
trivial lower bound

← LP takes long to solve

← little progress for much more time

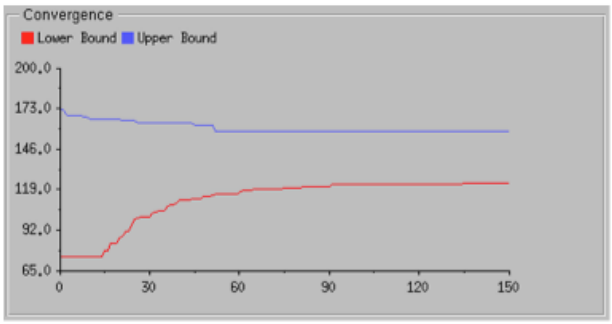
Sun Ultra 2, 200 MHz, LPs with CPLEX 6.5.3

We know from (the old) ADM II that the lower bound of the optimal Lagrange relaxation and the LP-relaxation are equal (in this case)
But the LP-relaxation is hard to solve

since the polyhedron of the simplified problem is integral

Computational results: upper bounds

600 instances with 120 jobs from PSPLIB



	CPU time	Above best solution
Our approach (≤ 220 iterations)	72.9 sec average 654 sec max	2.4 % average 9.5 % max

← documented in PSPLIB

Sun Ultra 2, 200 MHz, LPs with CPLEX 6.5.3

Comparison with other algorithms

genetic algorithm	Hartmann 1999	35.6
tabu search	Nonobe and Ibaraki	35.9
our approach		36.0
ant colony opt.	Merkle et al. 2000	36.7
constraint propagation	Dorndorf et al. 2000	37.1
simulated annealing	Bouleimen and Lecocq 2000	37.7
sampling	Kolisch 1996	38.7

← good and gives lower bounds!

deviation of best solution from critical path lower bound in %

feasible — trivial lower bound

Summary

- Iterative improvement of upper and lower bounds
- Both bounds (upper) comparable with state-of-the-art
- Drastically reduced computation times w.r.t. standard solvers
- Good tradeoff between quality of the bounds and computational effort
- Applicable to various extensions:
  - varying resource availability and consumption
  - general temporal constraints
  - other objective functions

Computational work today (highly active area)

Combine techniques from
- integer programming
- constraint programming
- SAT-solving

→ in SCIP

Jens Schulz Dissertation 2013