§21 Generalized Project Scheduling Problems and IP-Models

A. Generalizing precedence constraints by time lags

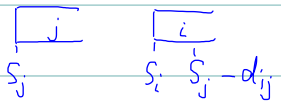
$i < j \iff S_i + x_i \le S_j$ $\qquad$ $S_i + d_{ij} \le S_j$

↑ positive processing time $\qquad$ ↑ may also be negative

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

Interpretation of negative d_{ij}:

$S_i + d_{ij} \le S_j \iff S_i \le S_j + (-d_{ij})$, with $(-d_{ij}) \ge 0$



$\iff$ S_i must start before $S_j - d_{ij}$ (a deadline)

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

↳ because of $S_i + x_i = C_i$

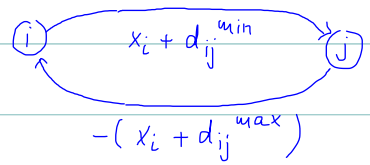
Therefore, time lags are often referred to as minimum time lags (lower bound) or maximum time lags (upper bound).

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

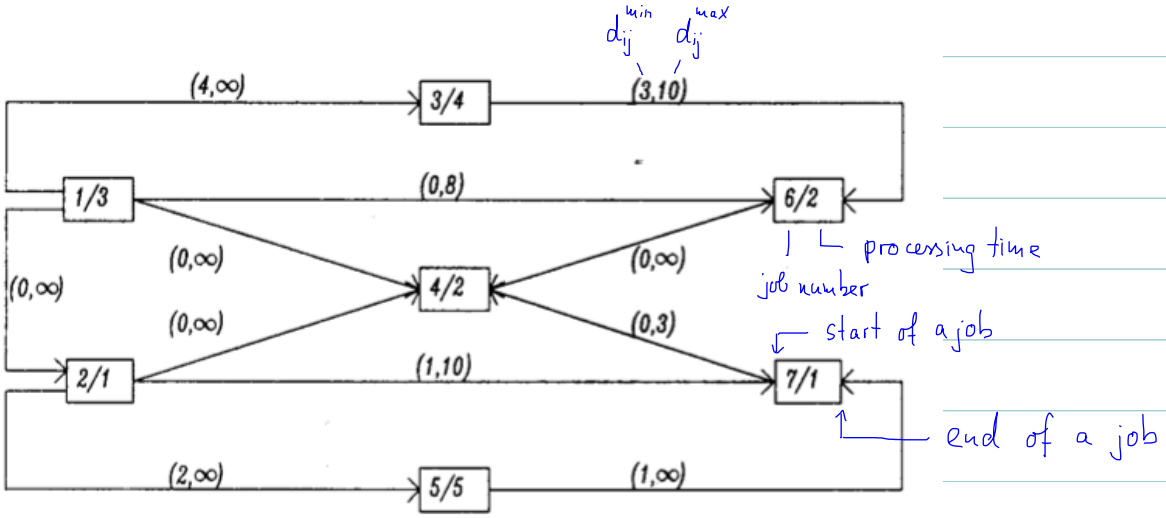
Expl: $C_i + d_{ij}^{min} \le S_j \le C_i + d_{ij}^{max}$ $\quad (d_{ij}^{min}, d_{ij}^{max} \ge 0)$ translates to

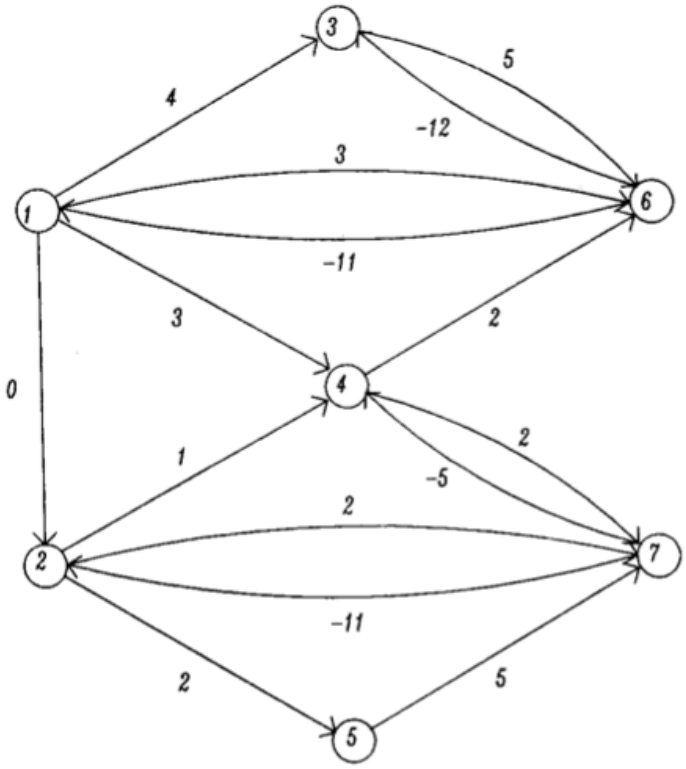
$S_i + x_i + d_{ij}^{min} \le S_j$ and $S_j - (x_i + d_{ij}^{max}) \le S_i$

$\Rightarrow$



Here is a larger example with different lags indicated by start/end of the arrows





The resulting digraph

It is clear that there is no feasible schedule if the digraph contains a directed cycle of positive length w.r.t. the d_{ij}. This is also sufficient.

21.1 Theorem:

a) A project network with time lags has a feasible schedule
$\Leftrightarrow$ 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) to j

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

Proof: see ADM I: existance of negative cycles, feasible potentials, and conservative arc weights $\square$

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

Assume $V = \{0, 1, \dots, n\}$ = set of jobs
(0: project start, n: project end, both with processing time 0)

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

$L \subseteq V \times V$ set of start-to-start normalized 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 cost of j

Formulation as IP

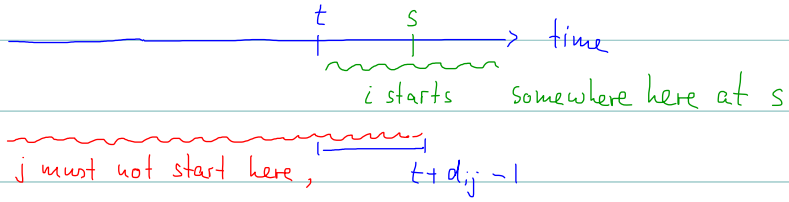
Variables: $x_{jt} := \begin{cases} 1 & \text{if job } j \text{ starts at time } t \\ 0 & \text{otherwise} \end{cases}$

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)

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

every time lag is respected



So if i starts at a and j at b and $a + d_{ij} > b$

then (3) is violated for $t = a$

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

We can compute for every job j in advance

- earliest start times $e(j) \geq 0$
- latest start times $l(j) \leq T - p_j$

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

Because of (2), we lift all w_{jt} by a constant M such that $\bar{w}_{jt} := w_{jt} + M \geq 0$

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 graph for the min-cut problem

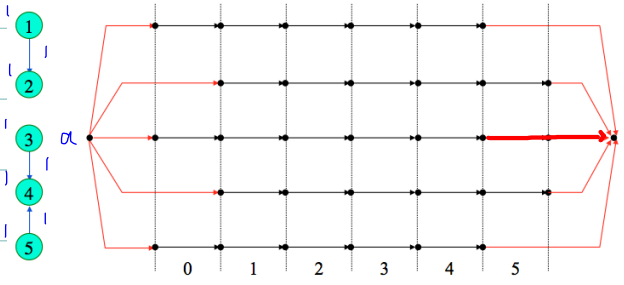
Nodes: one node v_{jt} for every job j and every possible start time t and $t+1$

$N = \{ v_{jt} \mid j \in V,\ t = e(j), \dots, \ell(j)+1 \} \cup \{a, b\}$ $\quad a$ = source, b = sink

Arcs: A = { assignment arcs } ∪ { temporal arcs } ∪ { auxiliary arcs }

assignment arcs = arcs $(v_{jt}, v_{j,t+1})$ $\forall\ j \in V$

auxiliary arcs = $(a, v_{j,e(j)})$ and $(v_{j,\ell(j)+1}, b)$ $\forall\ j \in V$



assignment arcs ⟶

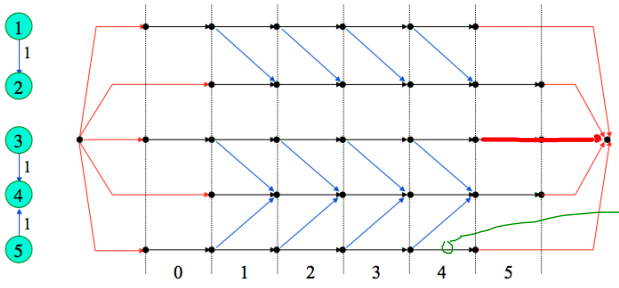
auxiliary arcs ⟶

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

$T = 6$

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

with $e(i) + 1 \le t \le \ell(i)$ and $e(j) + 1 \le t + d_{ij} \le \ell(j)$



temporal arcs ⟶

choice of this arc

$\hat{=}$ $x_{54} = 1$, i.e. $S_5 = 4$

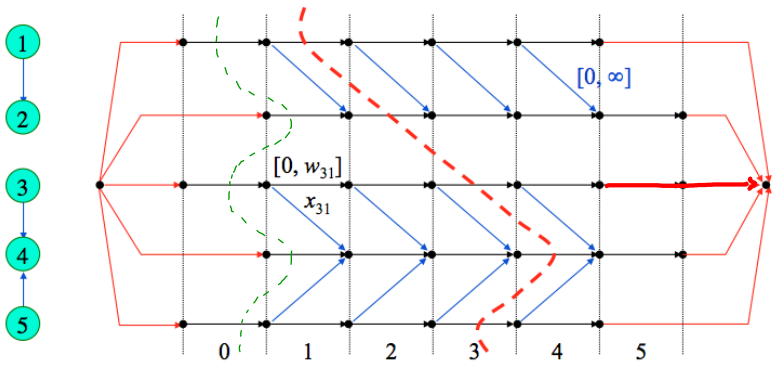
Arc capacities: assignment arcs: $c(v_{jt}, v_{j,t+1}) = w_{jt} \ge 0$ cost of starting job j at t

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

lower capacities of all arcs are 0

For our result, we are interested in special a,b-cuts of this graph.

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



defines an n-cut

defines the n-cut with earliest starts (that 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 there 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) $(X, \bar{X})$ contains at least one assignment arc for every job

For job j consider the "j-path" $a \to \underbrace{v_{j,s(t)} \to \dots \to v_{j,\ell(t)+1}}_{\text{all assignment arcs of } j} \to b$

This path must have a forward arc in $(X, \bar{X})$.

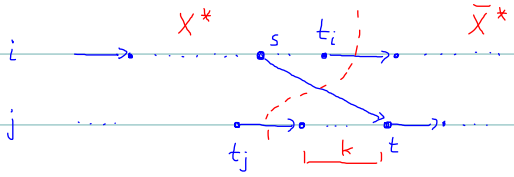
(2) If $(X, \bar{X})$ contains more than one assignment arc for some jobs, then we can construct from it an n-cut with the same capacity

For such jobs j let $t_j :=$ first index for which $v_{jt} \to v_{j,t+1}$ is forward arc in $(X, \bar{X})$

Define $X^* := \{a\} \cup \bigcup_j \{ v_{jt} \mid t \le t_j \}$

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

Suppose it does contain $v_{is} \to v_{jt}$ as forward arc $\hat{=}$ time lag $(i,j) \in L$, $d_{ij} = t - s$



$\hookrightarrow s \le t_i$, $t \ge t_j$

Let $k := t - (t_j + 1) \ge 0$

t_j is a possible start time for j

$\Rightarrow$ $e(j) + 1 \le t - k$ (i)

$s - k \le \ell(i)$ (ii)

↑ since $s - k \le s \le t_i \le \ell(i)$

$s - k \ge e(i) + 1$ (iii)

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

$\Rightarrow t - k = s - k + (t - s) < e(i) + 1 + (t - s)$

(i,j) time lag with $d_{ij} = t - s$ $\Rightarrow e(i) + (t - s) \le e(j)$

$\Rightarrow t - k < e(j) + 1$

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^* $\Rightarrow$ $v_{i,s-k} \in X$ (otherwise t_i would be different)

(i) $\Rightarrow$ $v_{j,t-k} = v_{j,t_j+1} \in \bar{X}$ (since the arc (v_{j,t_j}, v_{j,t_j+1}) is 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)

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

clear, since $(X^*, \bar{X}^*)$ contains fewer assignment arcs, no temporal arc, and $w_{jt} \ge 0$

(3) $(X^*, \bar{X}^*)$ can be constructed from $(X, \bar{X})$ in $O(nT)$ time

clear from the definition of X^* □

21.3 Theorem

There is a one-to-one correspondance between n-cuts $(X, \bar{X})$ of D with finite capacity and feasible solutions of project scheduling problem with start dependent cost (1)-(4), namely

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

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

21.4 Corollary

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

Proof of Theorem 21.3

(1) feasible solutions x of the IP (1)-(4) yield cuts $(X, \bar{X})$ with $w(x) = c(X, \bar{X})$

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

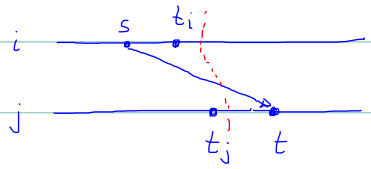
set $X := \bigcup_j \{ v_{jt} \mid t \le t_j \} \cup \{a\}$ $\Rightarrow$ $(X, \bar{X})$ is an n-cut

$\Rightarrow$ sum of capacities of assignment arcs in $(X, \bar{X})$ gives $w(x)$

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

$\Rightarrow$ this must be a temporal arc (v_{is}, v_{jt}), $s \le t_i$, $t > t_j$

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



$\Rightarrow$ $t_j - t_i < t - s = d_{ij}$, a contradiction to feasibility of x

$\Rightarrow$ $w(x) = c(X, \bar{X})$

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

follows directly from the definition in Theorem 21.3 □

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 B to this case

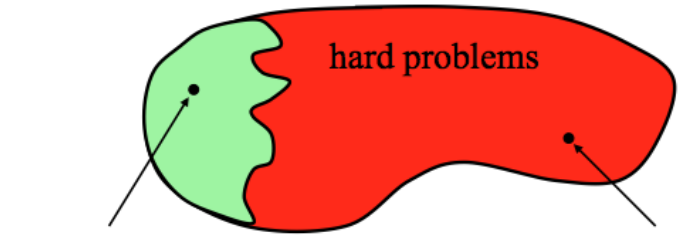
time-indexed binary variables $x_{jt} = \begin{cases} 1 : \text{job } j \text{ starts at time } t \\ 0 : \text{otherwise} \end{cases}$

$\min \sum_t t \cdot x_{n,t}$	minimize makespan	$\hat{=}$ (1), only job n (project completion) 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	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 does not exceed the available amount R_k

Simplify by relaxing the resource constraints



simplified problem — original problem, is very hard, see § 20

We apply Lagrangean 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

$$\sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+d_{ij}-1} x_{js} \le 1$$ — respect temporal distances d_{ij}

$$\sum_j r_{jk} \Big(\sum_{s=t-p_j+1}^{t} x_{js} \Big) \le R_k$$ — respect resource constraints

$$x_{jt} \in \{0, 1\}$$ — 0/1 variables

punish violation of capacity constraints in objective

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

group cost w.r.t. x_{jt}

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

$$\min \sum_{j,t} w_{jt} \cdot x_{jt} \qquad \text{cost for makespan and resource violations}$$

$$\sum_t x_{jt} = 1 \qquad \text{start every job } j$$

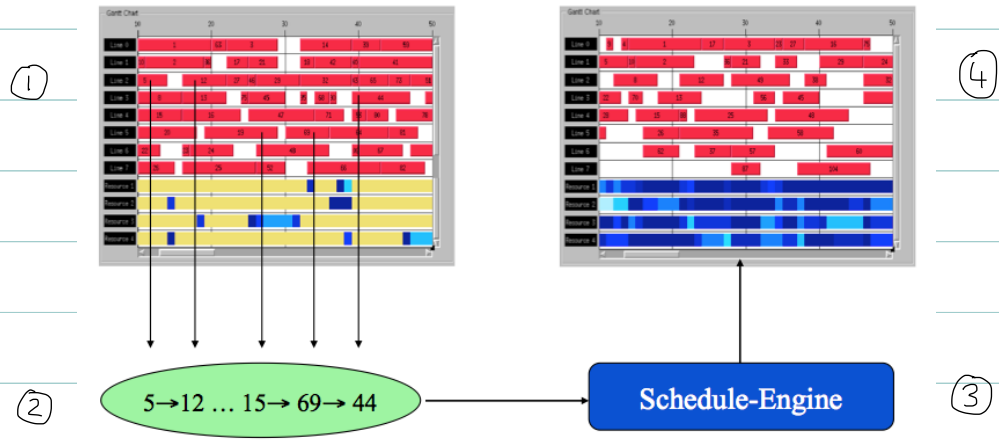
$$\sum_{s=t}^{T} x_{is} + \sum_{s=0}^{t+d_{ij}-1} x_{js} \leq 1 \qquad \text{respect temporal distances } d_{ij}$$

$$x_{jt} \in \{0, 1\} \qquad 0/1 \text{ variables}$$

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

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

From time-feasible to feasible schedules



Information ② is extracted from a time-feasible schedule ① and input to a schedule engine ③ that generates from ② a feasible schedule ④

The information ② is a list $i_1, i_2, \ldots, i_n$ with $S_{i_1} \le S_{i_2} \le \ldots \le S_{i_n}$ in ①

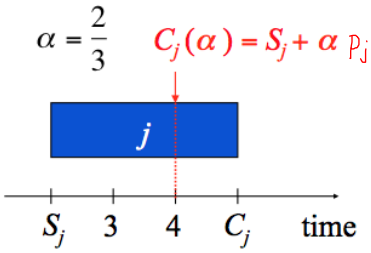
increasing start times

This list is input for the scheduling engine

Schedule-Engine

Ideas from various areas of Operations Research

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



sort jobs by increasing $C_j(\alpha)$

↑ used by us for generating the list for several values of α

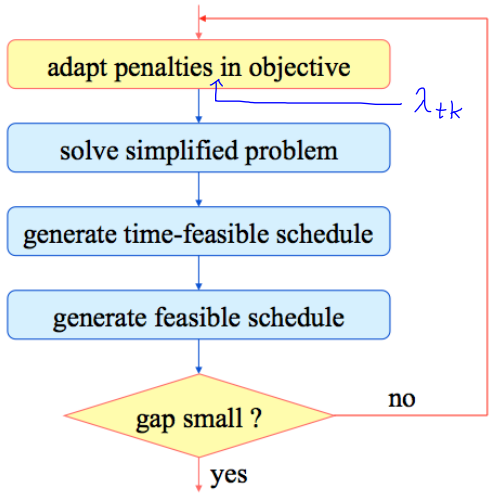
this gives different lists $L(\alpha)$

each such list is used for list scheduling (priority policy)

the best schedule is returned

Lagrange-relaxation and subgradient optimization

← combine with subgradient optimization for adapting the Lagrangean multipliers 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

Sun Ultra 2, 200 MHz, LPs with CPLEX 6.5.3

← LP takes long to solve

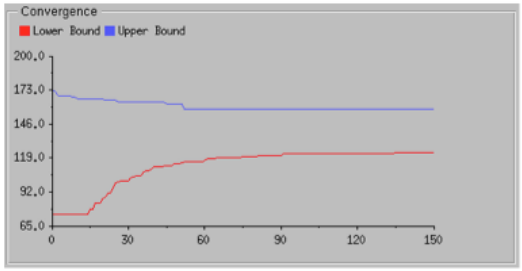
← little progress for much more time

We know from (the old) ADM II that (in this case) the lower bounds of the optimal Lagrangean relaxation and the LP-relaxation are equal

But the LP-relaxation is hard to solve!

Computational results: upper bounds

600 instances with 120 jobs from PSPLIB



still a large gap due to the intrinsic hardness of the RCPSP

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

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

other techniques have similar quality

↑

deviation of best solution from critical path lower bound in %

Summary

- Iterative improvement of upper and lower bounds
- Both bounds 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

Jens Schulz Dissertation 2012