

Scheduling

Zdenek Hanzalek
zdenek.hanzalek@cvut.cz

CTU in Prague

April 9, 2018

Scheduling - Basic Terminology

- **set of n tasks** $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$
- set of m types of resources (processors, machines, employees,...) with capacities R_k , $\mathcal{P} = \{P_1^1, \dots, P_{R_1}^1, P_1^2, \dots, P_{R_2}^2, \dots, P_1^m, \dots, P_{R_m}^m\}$
- **Scheduling is an assignment of a task to a resources in time**
- Each task must be **completed**
 - this differs from planning which decides which task will be scheduled and processed
- Set of tasks is known when executing the scheduling algorithm (this is called **off-line scheduling**)
 - this differs from on-line scheduling - OS scheduler, for example, schedules new tasks using some policy (e.g. priority levels)
- A result is a schedule which determines which task is run on which resource and when. Often depicted as a **Gantt chart**.

General and Specific Constraints

General constraints:

- Each task is to be processed **by at most one resource** at a time (task is sequential)
- Each resource is capable of processing **at most one task** at a time

Specific constraints:

- Task T_i has to be processed during time interval $\langle r_i, \tilde{d}_i \rangle$
- When the precedence constraint is defined between T_i and T_j , i.e. $T_i \prec T_j$, then the processing of task T_j can't start before task T_i was completed
- If scheduling is non-preemptive, a task cannot be stopped and completed later
- If scheduling is preemptive, the number of preemptions must be finite

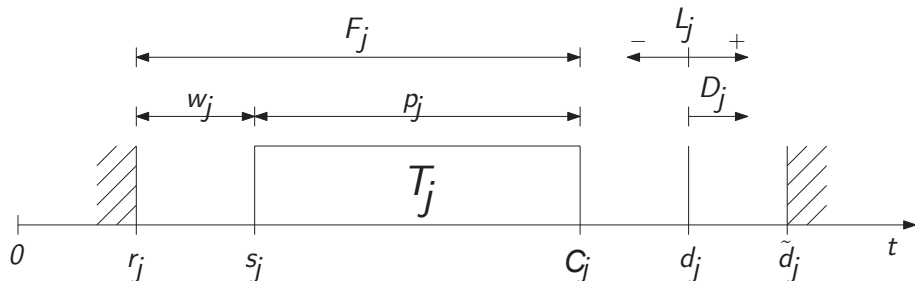
Task Parameters and Variables

Parameters

- **release time** r_j
- **processing time** p_j
- **due date** d_j , time in which task T_j should be completed
- **deadline** \tilde{d}_j , time in which task T_j has to be completed

Variables

- **start time** s_j
- **completion time** C_j
- waiting time $w_j = s_j - r_j$
- flow (lead) time $F_j = C_j - r_j$
- **lateness** $L_j = C_j - d_j$
- tardiness $D_j = \max\{C_j - d_j, 0\}$



Graham's Notation $\alpha | \beta | \gamma$

Classify scheduling problems by
resources | **tasks** | **criterion**

Example: $P2 | \text{pmtn} | C_{max}$ represents scheduling on two parallel identical resources, and preemption is allowed. The optimization criterion is the completion time of the last task.

α - **resources**

- **Parallel resources** - a task can run on any resource (only one type of resource exists with capacity R , i.e. $\mathcal{P} = \{P^1, \dots, P^R\}$).
- **Dedicated resources** - a task can run only on one resource (m resource types with unit capacity, i.e. $\mathcal{P} = \{P^1, P^2, \dots, P^m\}$).
- **Project Scheduling** - m resource types, each with capacity R_k , i.e. $\mathcal{P} = \{P_1^1, \dots, P_{R_1}^1, P_1^2, \dots, P_{R_2}^2, \dots, P_1^m, \dots, P_{R_m}^m\}$.

Resources Characteristics α_1, α_2

α_1	=	1	single resource
		P	parallel identical resources
		Q	parallel uniform resources, computation time is inversely related to resource speed
		R	parallel unrelated resources, computation times are given as a matrix (resources x tasks)
		O	dedicated resources - open-shop - tasks are independent
		F	dedicated resources - flow-shop - tasks are grouped in the sequences (jobs) in the same order, each job visits each machine once
		J	dedicated resources - job-shop - order of tasks in jobs is arbitrary, resource can be used several times in a job
		PS	Project Scheduling - most general (several resource types with capacities, general precedence constraints)
α_2	=	\emptyset	arbitrary number of resources
		2	2 resources (or other specified number)
		m, R	m resource types with capacities R (Project Scheduling)

Task Characteristics $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8$

β_1	= pmtn \emptyset	preemption is allowed preemption is not allowed
β_2	= prec in-tree,out-tree chain tmpn \emptyset	precedence constraints tree constraints chain constraints temporal constraints (for Project Sched.) independent tasks
β_3	= r_j	release time
β_4	= $p_j = k$ $p_L \leq p_j \leq p_U$ \emptyset	uniform processing time restricted processing time arbitrary processing time
β_5	= \tilde{d}_j, d_j	deadline, due-date
β_6	= $n_j \leq k$	maximal number of tasks in a job
β_7	= no-wait	buffers of zero capacity
β_8	= set-up	time for resource reconfiguration

Optimality Criterion γ

$\gamma = C_{max}$	minimize schedule length $C_{max} = \max \{C_j\}$ (makespan, i.e. completion time of the last task)
$\sum C_j$	minimize the sum of completion times
$\sum w_j C_j$	minimize weighted completion time
L_{max}	minimize max. lateness $L_{max} = \max \{C_j - d_j\}$
\emptyset	decision problem
...	

An Example: $P \parallel C_{max}$ means:

Scheduling on an arbitrary number of parallel identical resources, no preemption, independent tasks (no precedence), tasks arrive to the system at time 0, processing times are arbitrary, objective is to minimize the schedule length.

Scheduling on One Resource

Minimizing Makespan (i.e. schedule length C_{max})

- $1 |prec| C_{max}$ - easy
 - the tasks are processed in an arbitrary order that satisfies the precedence relation (i.e. topological order), $C_{max} = \sum_{j=1}^n p_j$
- $1 || C_{max}$ - easy
- $1 |r_j| C_{max}$ - easy
 - the tasks are processed in a non-descending order of r_j (T_j with the lowest r_j first)
- $1 |\tilde{d}_j| C_{max}$ - easy
 - the tasks are processed in a non-descending order of \tilde{d}_j
 - can be solved by EDF - Earliest Deadline First
 - the feasible schedule doesn't have to exist
- $1 |r_j, \tilde{d}_j| C_{max}$ - NP-hard
 - NP-hardness proved by the pol. reduction from 3-Partition problem
 - for $p_j = 1$ there exists a polynomial algorithm

1 $\left| r_j, \tilde{d}_j \right| C_{max}$ Problem is NP-hard

Theorem

The 1 $\left| r_j, \tilde{d}_j \right| C_{max}$ problem is NP-hard in the strong sense.

By reduction from the 3-Partition problem, which is strongly NP-complete.

3-Partition decision problem instance, $I_{3P} = (A, B)$, is given as:

- a multiset A of $3m$ integers a_1, a_2, \dots, a_{3m} (sizes of **items**), and
- a positive integer B (size of **bins**) such that
 $\forall i \in \{1, 2, \dots, 3m\} : \frac{B}{4} < a_i < \frac{B}{2}$ and $\sum_{i=1}^{3m} a_i = mB$.

The problem is to determine whether A **can be partitioned** into m disjoint subsets A_1, A_2, \dots, A_m such that, $\forall j \in \{1, 2, \dots, m\} : \sum_{a_i \in A_j} a_i = B$.

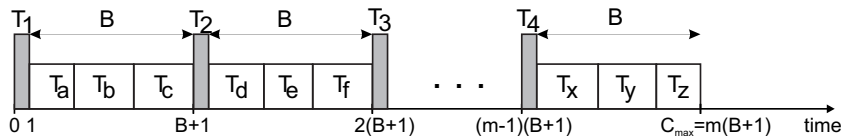
Note: if we show that there is a subset A_j which contains integers summing to B , then it must contain **three integers**. This follows from the assumption $\frac{B}{4} < a_i < \frac{B}{2}$ (try to sum-up 4 integers or 2 integers).

Reduction from 3-Partition to $1 \mid r_j, \tilde{d}_j \mid C_{max}$

From the given instance of the 3-Partition problem $I_{3P} = (A, B)$, we build $1 \mid r_j, \tilde{d}_j \mid C_{max}$ scheduling problem instance I_{SCH} comprised of $4m$ tasks

$T_j = (p_j, r_j, \tilde{d}_j)$ as follows:

- $\forall j \in \{1, \dots, m\} : T_j = (1, (B+1) \cdot (j-1), (B+1) \cdot (j-1)+1)$. These are “additional/artificial” tasks used to separate the subsets.
- $\forall j \in \{m+1, \dots, 4m\} : T_j = (a_i, 0, \infty), i = j - m$.
Each of these tasks T_j corresponds to the element a_i of I_{3P} .



It is easy to prove that the $I_{3P} = (A, B)$ has a solution if and only if the optimal solution of the related I_{SCH} has value of $C_{max} = m \cdot (B+1)$.

Position based ILP formulation for $1 \mid r_j, \tilde{d}_j \mid C_{max}$

$x_{iq} = 1$ iff task i is at the q -th position in the sequence of tasks

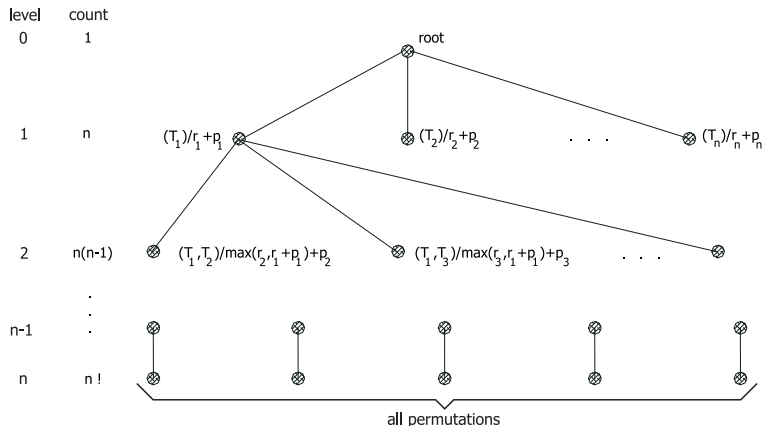
$$\begin{aligned}
 & \min C_{max} \\
 & \text{subject to:} \\
 & \sum_{q=1}^n x_{iq} = 1 && i = 1..n \\
 & \sum_{i=1}^n x_{iq} = 1 && q = 1..n \\
 & t_q \geq \sum_{i=1}^n r_i \cdot x_{iq} && q = 1..n \\
 & t_q \geq t_{q-1} + \sum_{i=1}^n p_i \cdot x_{i,q-1} && q = 2..n \\
 & t_q \leq \sum_{i=1}^n \tilde{d}_i \cdot x_{i,q} - \sum_{i=1}^n p_i \cdot x_{i,q} && q = 1..n \\
 & C_{max} \geq t_n + \sum_{i=1}^n p_i \cdot x_{in}
 \end{aligned}$$

variables: $x_{i \in 1..n, q \in 1..n} \in \{0, 1\}$, $C_{max} \in \langle 0, UB \rangle$, $t_{q \in 1..n} \in \langle 0, UB \rangle$

Bratley's Algorithm for $1 \mid r_j, \tilde{d}_j \mid C_{max}$

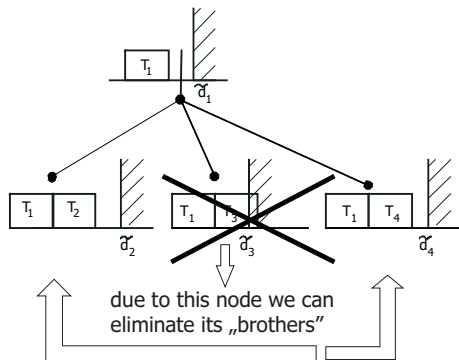
A **branch and bound** (B&B) algorithm.

Branching - without bounding it is an **enumerative method** that creates a solution tree (some of the nodes are infeasible). Every node is labeled by: (the order of tasks)/(completion time of the last task).



(i) **eliminate the node** exceeding the deadline (and all its “brothers”)

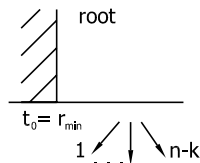
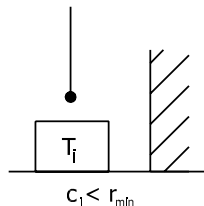
- If there is a node which exceeds any deadline, all its descendants should be eliminated
- Critical task (here T_3) will have to be scheduled anyway - therefore, all of its “brothers” should be eliminated as well



Tree Size Reduction - Decomposition

(ii) **problem decomposition** due to idle waiting - e.g. when the employee waits for the material, his work was optimal

- Consider node v on level k . If C_i of the last scheduled task is less than or equal to r_i of all unscheduled tasks, there is no need for backtrack above v
- v becomes a new root and there are $n - k$ levels ($n - k$ unscheduled tasks) to be scheduled



Optimality Test - Termination of Bratley's Algorithm

Definition: BRTP - Block with Release Time Property

BRTP is a set of k tasks that satisfy:

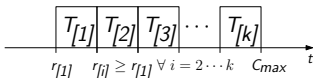
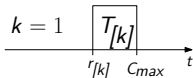
- first task $T_{[1]}$ starts at its release time
- all k tasks till the end of the schedule run without “idle waiting”
- $r_{[1]} \leq r_{[i]}$ for all $i = 2 \dots k$

Note: “till the end of the schedule” implies there is at most one BRTP

Lemma: sufficient condition of optimality

If BRTP exists, the schedule is optimal (the search is finished).

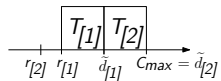
Proof:



- this schedule is optimal since the last task $T_{[k]}$ can not be completed earlier
- order of prec. tasks is not important - see (ii)
- no task from BRTP can be done before $r_{[1]}$
- there is no task after C_{max}

B RTP is not Necessary Condition of Optimality

Example:



In this particular case, the schedule is optimal, but it does not have B RTP.

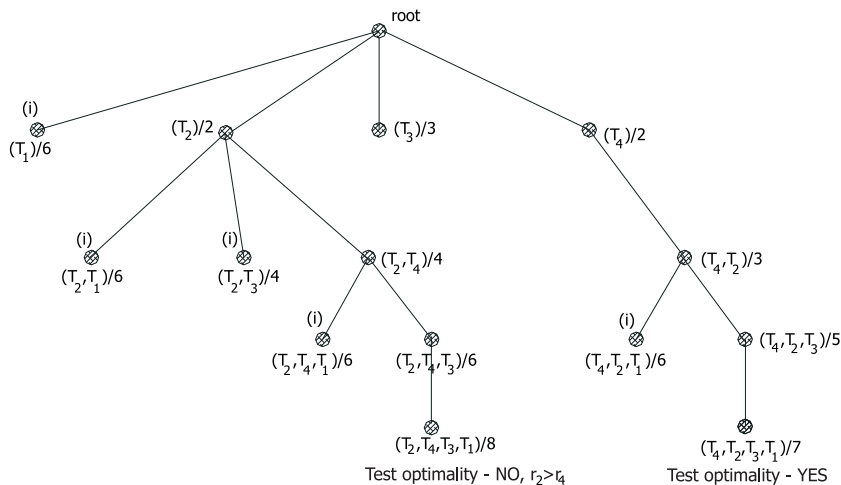
Tightening the bounds:

In general, C_{max} found without B RTP could be used for bounding further solutions while **setting all deadlines** to be at most $C_{max} - \epsilon$.

This ensures that if other feasible schedules exist, only those that are **better by** ϵ than the solution at hand, are generated.

Bratley's Algorithm - Example

$$r = (4, 1, 1, 0), \quad p = (2, 1, 2, 2), \quad \tilde{d} = (8, 5, 6, 4)$$



Scheduling on One Resource

Minimizing $\sum w_j C_j$

- $1 \parallel \sum C_j$ - easy
 - SPT rule (Shortest Processing Time first) - schedule the tasks in a non-decreasing order of p_j
- $1 \parallel \sum w_j C_j$ - easy
 - Weighted SPT - schedule the tasks in a non-decreasing order of $\frac{p_j}{w_j}$
- $1 |r_j| \sum C_j$ - NP-hard
- $1 |pmtn, r_j| \sum C_j$ - can be solved by modified SPT
- $1 |pmtn, r_j| \sum w_j C_j$ - NP-hard
- $1 \left| \tilde{d}_j \right| \sum C_j$ - can be solved by modified SPT
- $1 \left| \tilde{d}_j \right| \sum w_j C_j$ - NP-hard
- $1 |prec| \sum C_j$ - NP-hard

First, we formulate the problem as an ILP:

- we use **variable** $x_{ij} \in \{0, 1\}$ such that $x_{ij} = 1$ iff T_i precedes T_j or $i = j$
- we encode **precedence relations** into $e_{ij} \in \{0, 1\}$ such that $e_{ij} = 1$ iff there is a directed edge from T_i to T_j in the precedence graph G or $i = j$
- **criterion** - completion time of task T_j consists of p_j and the processing time of its predecessors:

$$\begin{aligned}
 C_j &= \sum_{i=1}^n p_i \cdot x_{ij} \\
 w_j \cdot C_j &= \sum_{i=1}^n p_i \cdot x_{ij} \cdot w_j \\
 J = \sum_{j=1}^n w_j \cdot C_j &= \sum_{j=1}^n \sum_{i=1}^n p_i \cdot x_{ij} \cdot w_j
 \end{aligned}$$

from all feasible schedules x we look for the one that minimizes $J(x)$,
i.e. $\min_x J(x)$

ILP formulation for $1 \mid \text{prec} \mid \sum w_j C_j$

$$\begin{aligned} \min \quad & \sum_{j=1}^n \sum_{i=1}^n p_i \cdot x_{ij} \cdot w_j \\ \text{subject to:} \quad & x_{i,j} \geq e_{i,j} \quad i, j \in 1..n && \text{if } T_i \text{ precedes } T_j \text{ in } G, \\ & && \text{then it precedes } T_j \\ & && \text{in the schedule} \\ & x_{i,j} + x_{j,i} = 1 \quad i, j \in 1..n, i \neq j && \text{either } T_i \text{ precedes } T_j, \\ & && \text{or vice versa} \\ & 1 \leq x_{i,j} + x_{j,k} + x_{k,i} \leq 2 \quad i, j, k \in 1..n, && \text{no cycle exists in the} \\ & && i \neq j \neq k && \text{digraph of } x \\ & x_{i,i} = 1 \quad i \in 1..n \end{aligned}$$

parameters: $w_{i \in 1..n}, p_{i \in 1..n} \in \mathbb{R}_0^+$ $e_{i \in 1..n, j \in 1..n} \in \{0, 1\}$

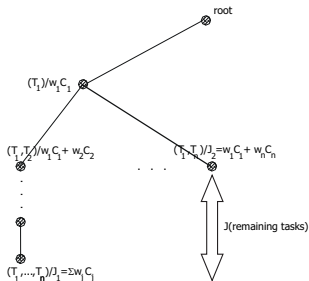
variables: $x_{i \in 1..n, j \in 1..n} \in \{0, 1\}$

Branch and Bound with LP Bounding

We relax on the integrality of variable x :

- $0 \leq x_{ij} \leq 1$ and $x_{i \in 1..n, j \in 1..n} \in \mathbb{R}$
- This does not give us the right solution, however we can use the J^{LP} (remaining tasks) value of this LP formulation as a lower bound on the “amount of remaining work”

The Branch and Bound algorithm creates a similar tree as Bradley’s algorithm.



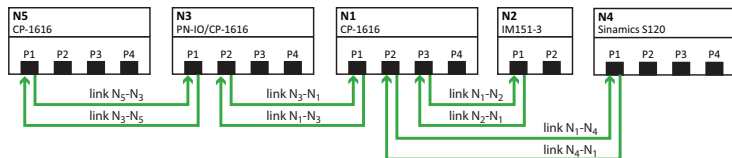
- Let J_1 be the value of the best solution known up to now
- We discard the partial solution of value J_2 not only when $J_2 \geq J_1$, but also when $J_2 + J^{LP}$ (remaining tasks) $\geq J_1$. Since the solution space of ILP is a subspace of LP we know:
 J (remaining tasks) $\geq J^{LP}$ (remaining tasks).

- $PS1 | \text{temp} | C_{max}$ - NP-hard
 - $PS1$ stands for single resource, temp stands for temporal constraints
 - Input: The number of non-preemptive tasks n and processing times (p_1, p_2, \dots, p_n) . The temporal constraints defined by digraph G .
 - Output: n -element vector s , where s_i is the start time of T_i
 - We will show Time-indexed and Relative-order ILP formulations
- $PSm, 1 | \text{temp} | C_{max}$ - NP-hard
 - $PSm, 1$ stands for m resource types, each of capacity 1
 - Input: The number of non-preemptive tasks n and processing times (p_1, p_2, \dots, p_n) . The temporal constraints defined by digraph G .
The number of dedicated resources m and the assignment of the tasks to the resources (a_1, a_2, \dots, a_n) .
 - Output: n -element vector s , where s_i is the start time of T_i
 - We show the Relative-order ILP formulation
- extensions to general scheduling model
 - multi-processor tasks
 - sequence dependent set-up times

Motivation Example: Message Scheduler for Profinet IO IRT - Specification

Profinet IO IRT is an Ethernet-based hard-real time communication protocol, which uses static schedules for time-critical data. Each node contains a special hardware switch that intentionally breaks the standard forwarding rules for a specified part of the period to ensure that no queuing delays occur for time-critical data.

Goal: Minimize the makespan (the schedule length) for time critical messages.



line	$N_1 \rightarrow N_3$	$N_1 \rightarrow N_4$	$N_1 \rightarrow N_2$	$N_2 \rightarrow N_1$	$N_3 \rightarrow N_1$	$N_4 \rightarrow N_1$	$N_3 \rightarrow N_5$	$N_5 \rightarrow N_3$
line delay [ns]	4875	5130	5862	3841	4875	4895	4875	4875

Motivation Example: Message Scheduler for Profinet IO IRT - Specification

Constraints:

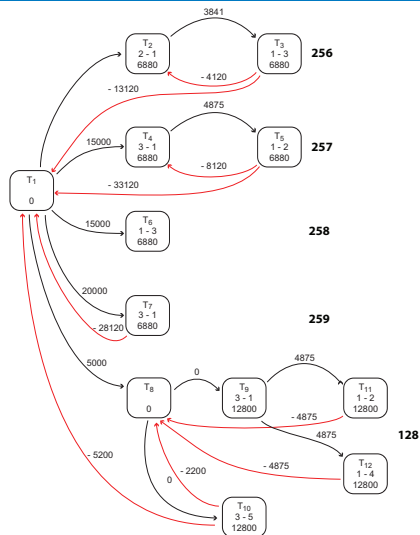
- tree topology \Rightarrow fixed routing
- release date r - earliest time the message can be sent
- deadline \tilde{d} - latest time the message can be delivered
- maximal allowed end-to-end time delay

ID	source \rightarrow target	length [ns]	r [ns]	\tilde{d} [ns]	end2end delay [ns]
256	$N_2 \rightarrow N_3$	5760	5000	20000	11000
257	$N_3 \rightarrow N_2$	5760	15000	40000	15000
258	$N_1 \rightarrow N_3$	5760	15000	-	-
259	$N_3 \rightarrow N_1$	5760	20000	35000	-
128	$N_3 \rightarrow \{N_1, N_2, N_4, N_5\}$	11680	5000	$\{-, -, -, 18000\}$	$\{-, 17675, 17675, 15000\}$

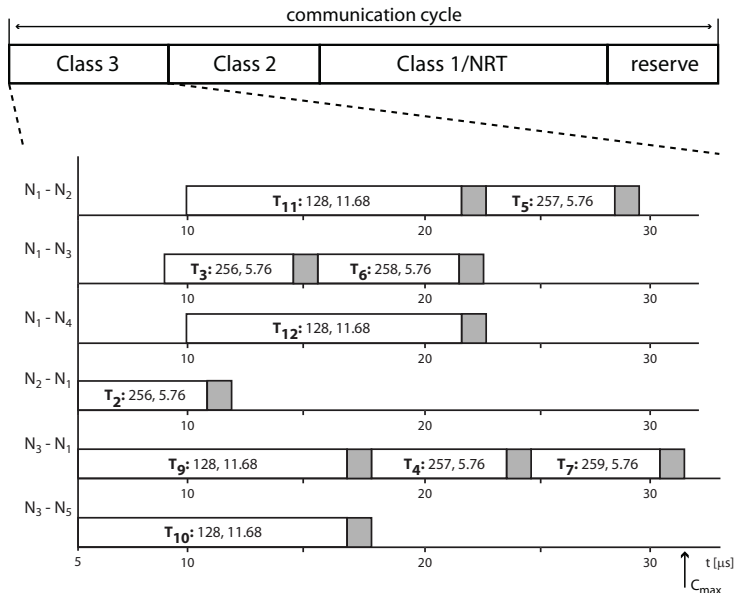
Motivation Example: Message Scheduler for Profinet IO IRT - Formalization

Can be formulated as
 $PSm, 1 | temp | C_{max}$ problem.

- task = message on a given line
- positive cost edge = r , precedence relations
- negative cost edge = \tilde{d} , end-to-end delay
- unicast message = chain of tasks (assuming positive edges)
- multicast message = out-tree of tasks (assuming positive edges)

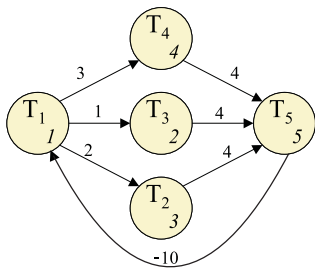


Motivation Ex.: Message Sch. for Profinet IO IRT - Result

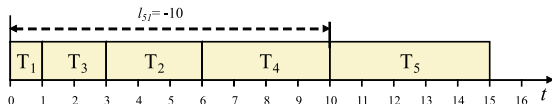


Temporal Constraints

- Set of non-preemptive tasks $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$ is represented by the nodes of the directed graph G (may include negative cycles).
- Processing time p_i is assigned to each task.



- The edges represent temporal constraints. Each edge from T_i to T_j has the length l_{ij} .
- Each temporal constraint is characterized by one inequality $s_i + l_{ij} \leq s_j$.



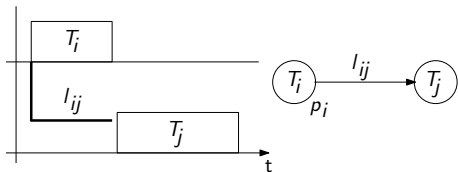
Temporal Constraints $s_i + l_{ij} \leq s_j$ with Positive l_{ij}

Temporal Constraints (also called a **generalized precedence constraint** or a **positive-negative time lag**)

- the start time of one task depends on the start time of another task

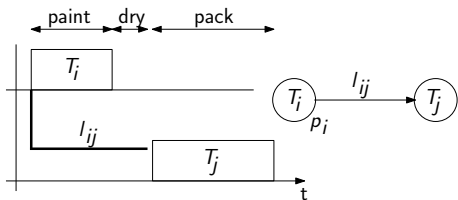
a) $l_{ij} = p_i$

- “normal” precedence relation
- the second task can start when the previous task is finished



b) $l_{ij} > p_i$

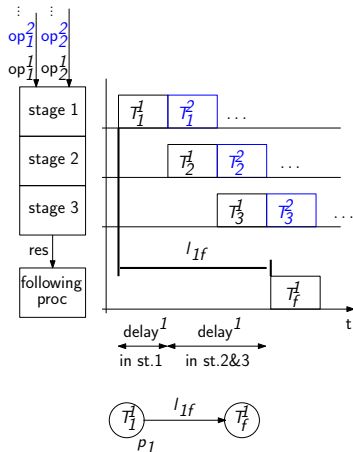
- the second task can start some time after the completion of previous task
- b.1) example of a dry operation performed in sufficiently large space



Temporal Constraints $s_i + l_{ij} \leq s_j$ with Positive l_{ij}

b.2) another example with $l_{ij} > p_i$ - pipe-lined ALU

- We assume the processing time to be equal in all stages
- **Result is available** l_{1f} tics after stage 1 reads operands
- **Stage 1 reads new operands** each p_1 tics
- **Stages 2 and 3 are not modeled** since we have enough of these resources and they are synchronized with stage 1



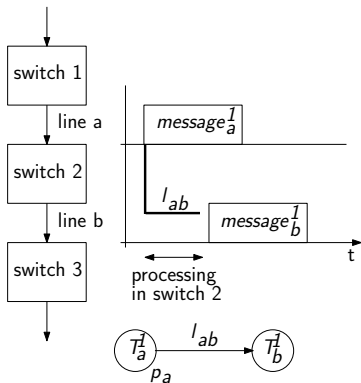
Temporal Constraints $s_i + l_{ij} \leq s_j$ with Positive l_{ij}

c) $0 < l_{ij} < p_i$

Partial results of the previous task may be used to start the execution of the following task.

E.g. the **cut-through** mechanism, where the switch starts transmission on the output port earlier than it receives the complete message on the input port.

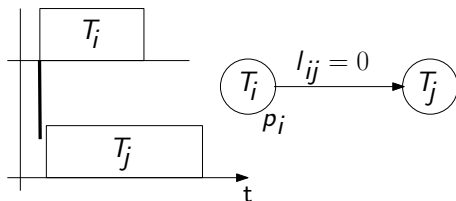
- time-triggered protocol
- **resources are communication links**
- l_{ab} represents the **delay in the switch**
- **different parts** of the same message are transmitted by several communication links at the same time



Temporal Constraints $s_i + l_{ij} \leq s_j$ with Zero or Negative l_{ij}

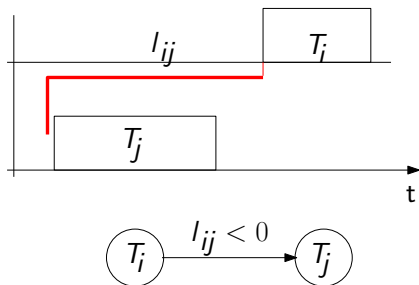
d) $l_{ij} = 0$

- Task T_i has to start earlier or at the same time as T_j



e) $l_{ij} < 0$

- Task T_i has to start earlier or **at most** $|l_{ij}|$ **later** than T_j
- It loses the sense of “normal” precedence relation, since T_i does not have to precede T_j
- It represents the **relative deadline** of T_i related to the start-time of T_j



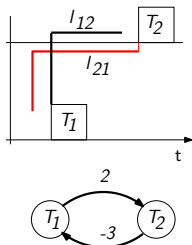
Cycles and Relative Time Windows

Absence of a positive cycle in graph G

- is a **necessary condition** for schedulability of $PS1 | \text{temp} | C_{max}$
- is a **necessary and sufficient condition** for schedulability of the instance with **unlimited capacity of resources**. The schedule, which is restricted only by the temp. constraints, can be found in pol. time
 - by LP or
 - by the longest paths. For G we can create G' , a **complete digraph of longest paths**, where weight l_{ij} is the length of the longest directed path from T_i to T_j in G (if no directed path in G exists, the weight is $l_{ij} = -\infty$). A start time of T_j is lower bounded by the longest path from arbitrary node, i.e. $s_j \geq \max_{vi \in 1 \dots n} l_{ij}$.

Example - relative time window, e.g. when applying a catalyst to the chemical process
If finite $l_{ij} \geq 0$ and $l_{ji} < 0$ do exist, tasks T_i and T_j are constrained by the relative time window.

- the length of the negative cycle determines the “clearance” of the time window



Task can be represented in two ways:

- **Time-indexed** - ILP model is based on variable x_{it} , which is equal to 1 iff $s_i = t$. Otherwise, it is equal to zero. Processing times are assumed to be positive integers.
- **Relative-order** - ILP model is based on the relative order of tasks given by variable x_{ij} , which is equal to 1 iff task T_i precedes task T_j . Otherwise, it is equal to zero. The processing times are nonnegative real numbers (tasks with zero processing time may be used to represent events).

Both models contain two types of constraints:

- temporal constraints
- resource constraints - prevent overlapping of tasks

min C_{max}

$$\sum_{t=0}^{UB-1} (t \cdot x_{it}) + l_{ij} \leq \sum_{t=0}^{UB-1} (t \cdot x_{jt}) \quad \forall l_{ij} \neq -\infty \text{ a } i \neq j \text{ (temp. const.)}$$

$$\sum_{i=1}^n \left(\sum_{k=\max(0, t-p_i+1)}^t x_{ik} \right) \leq 1 \quad \forall t \in \{0, \dots, UB-1\} \text{ (resource)}$$

$$\sum_{t=0}^{UB-1} x_{it} = 1 \quad \forall i \in \{1, \dots, n\} \text{ (} T_i \text{ is scheduled)}$$

$$\sum_{t=0}^{UB-1} (t \cdot x_{it}) + p_i \leq C_{max} \quad \forall i \in \{1, \dots, n\}$$

variables: $x_{it} \in \{0, 1\}$, $C_{max} \in \{0, \dots, UB\}$

UB - upper bound of C_{max} (e.g. $UB = \sum_{i=1}^n \max \{p_i, \max_{i,j \in \{1, \dots, n\}} l_{ij}\}$).

Start time of T_i is $s_i = \sum_{t=0}^{UB-1} (t \cdot x_{it})$.

Model contains $n \cdot UB + 1$ variables and $|E| + UB + 2n$ constraints.

Constant $|E|$ represents the number of temporal constraints (edges in G).

Time-indexed Model for $PS1$ |temp| C_{max}

$$\mathcal{T} = \{T_1, T_2, T_3\}, \rho = (1, 2, 1), UB = 5$$

T_1 is scheduled:

	0	1	2	3	4
T_1	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
T_2	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}
T_3	x_{30}	x_{31}	x_{32}	x_{33}	x_{34}

$\Sigma = 1$

Resource constr. at time 2:

	0	1	2	3	4
T_1	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
T_2	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}
T_3	x_{30}	x_{31}	x_{32}	x_{33}	x_{34}

$\Sigma \leq 1$

Resource constraint for couple of tasks:

$$p_j \leq s_i - s_j + UB \cdot x_{ij} \leq UB - p_i$$

The constraint uses “big M” (here UB - upper bound on C_{max}).

If $x_{ij} = 1$, T_i **precedes** task T_j and the constraint is formulated as $s_i + p_i \leq s_j$.

If $x_{ij} = 0$, T_i **follows** task T_j and the constraint is formulated as $s_j + p_j \leq s_i$.

Relative-order Model for $PS1 \mid temp \mid C_{max}$

$$\min C_{max}$$

$$s_i + l_{ij} \leq s_j \quad \forall l_{ij} \neq -\infty \text{ and } i \neq j$$

(temporal constraint)

$$p_j \leq s_i - s_j + UB \cdot x_{ij} \leq UB - p_i \quad \forall i, j \in \{1, \dots, n\} \text{ and } i < j$$

(resource constraint)

$$s_i + p_i \leq C_{max} \quad \forall i \in \{1, \dots, n\}$$

variables: $x_{ij} \in \{0, 1\}$, $C_{max} \in \langle 0, UB \rangle$, $s_i \in \langle 0, UB - p_i \rangle$

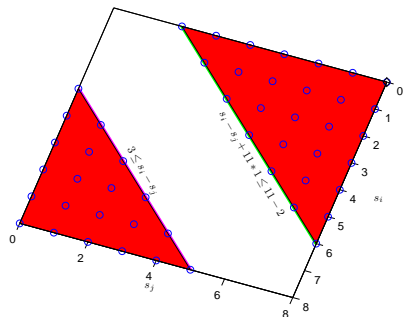
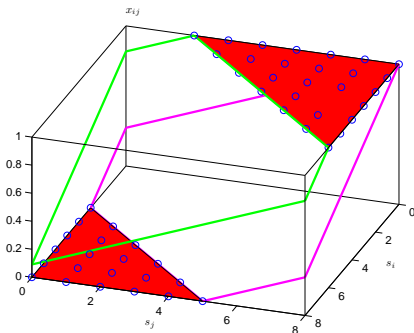
The model contains $n + (n^2 - n) / 2 + 1$ variables
and $|E| + (n^2 - n) + n$ constraints.

$|E|$ is a number of temporal constraints (edges in G).

Relative-order Model for $PS1 | \text{temp} | C_{max}$

Example: no temporal constraints, two tasks T_i, T_j with $p_i = 2$ and $p_j = 3$. We set $UB = 11$ and we study $s_i \in \langle 0, 8 \rangle$.

3D polytope (left) is determined by the resource constr. given by blue and green hyperplanes (see colors on the previous slide). Its projection to 2D space (right) shows both sequences of tasks. When we change UB , the hyperplanes in 3D decline -each of the moves the vertex with acute angle.



Comparison of the Two Models

Each model is suitable for different types of tasks:

Time-indexed model:

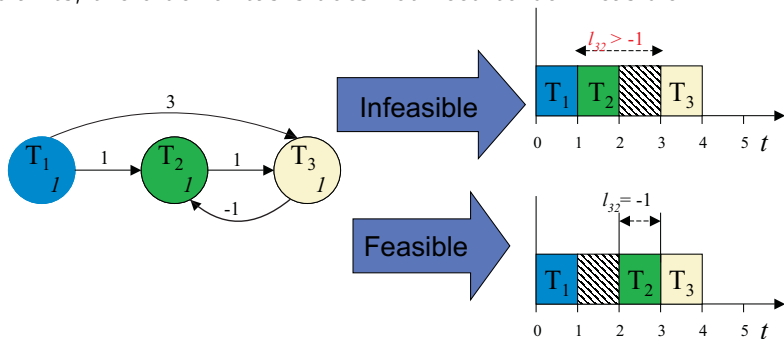
- (+) Can be easily extended for parallel identical processors.
- (+) ILP formulation does not need many constraints.
- (-) The size of the model grows with the size of UB .

Relative-order model:

- (+) The size of ILP model does not depend on UB .
- (-) Requires a big number of constraints.

Feasibility Test for Heuristic Algorithms

If the partial schedule (found for example by a greedy algorithm which inserts tasks in a topological order of edges with positive weight, or the partial result during the Branch and Bound algorithm) violates some time constraints, the order of tasks does not need to be infeasible.



When the optimal order of the tasks in the schedule is known (variables x_{ij} are constants), it is easy to find the start time of the tasks (for example by LP formulation involving time constraints only).

Relative-order Model for Project Scheduling with Dedicated Resources of Unit Capacity $PSm, 1 | \text{temp} | C_{max}$

Part of the input parameters are the number of resources m and **assignment of the tasks to the resources** $(a_1, \dots, a_i, \dots, a_n)$, where a_i is index of the resource type on which task T_i will be running.

$$\min C_{max}$$

$$s_i + l_{ij} \leq s_j \quad \forall l_{ij} \neq -\infty \text{ and } i \neq j$$

(temporal constraints)

$$p_j \leq s_i - s_j + UB \cdot x_{ij} \leq UB - p_i \quad \forall i, j \in \{1, \dots, n\}, i < j \text{ and } \underline{a_i = a_j}$$

(on the same resource type)

$$s_i + p_i \leq C_{max} \quad \forall i \in \{1, \dots, n\}$$

variables: $x_{ij} \in \{0, 1\}$, $C_{max} \in \langle 0, UB \rangle$, $s_i \in \langle 0, UB \rangle$

Model consists of less than $n + (n^2 - n) / 2 + 1$ variables (exact number depends on the number of tasks scheduled on each resource type).

Modeling with Temporal Constraints

Using $PS1 | \text{temp} | C_{max}$ we will model:

- $1 | r_j, \tilde{d}_j | C_{max}$
- scheduling on **dedicated resources** $PSm, 1 | \text{temp} | C_{max}$

Using $PSm, 1 | \text{temp} | C_{max}$ we will model:

- scheduling of **multiprocessor task** - task needs more than one resource type at a given moment,
- scheduling with **setup times** - two subsequent tasks executed on one resource need to be separated by idle waiting, for example to change the tool.

Reduction from $1 \mid r_j, \tilde{d}_j \mid C_{max}$ to $PS1 \mid \text{temp} \mid C_{max}$

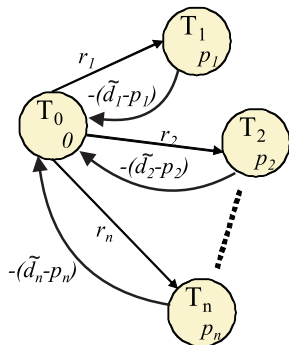
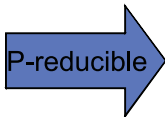
This polynomial reduction proves that $PS1 \mid \text{temp} \mid C_{max}$ is NP-hard, since Bratley's problem is NP-hard.

Instance $1 \mid r_j, \tilde{d}_j \mid C_{max}$

$$r = (r_1, r_2, \dots, r_n)$$

$$p = (p_1, p_2, \dots, p_n)$$

$$\tilde{d} = (\tilde{d}_1, \tilde{d}_2, \dots, \tilde{d}_n)$$



Reduction from $PSm, 1 | \text{temp} | C_{max}$ to $PS1 | \text{temp} | C_{max}$ is based on the projection of **each resource to the independent time window**. In other words, the schedule of tasks on P^j is projected into interval $\langle (j-1) \cdot UB, j \cdot UB \rangle$

Transformation consists of two steps:

- **Add dummy tasks** T_0 and T_{n+1} with $p_0 = p_{n+1} = 0$.
 - Task T_0 , processed on P^1 , precedes all tasks $T_i \in \mathcal{T}$, ie. $s_0 \leq s_i$.
 - Task T_{n+1} , processed on P^m , follows all task $T_i \in \mathcal{T}$, tj. $s_i + p_i \leq s_{n+1}$.
- **Transform the original temporal constraints** to $l'_{ij} = l_{ij} + (a_j - a_i) \cdot UB$.

The new start time s'_i of each task on processor a_i is:
 $s'_i = s_i + (a_i - 1) \cdot UB$.

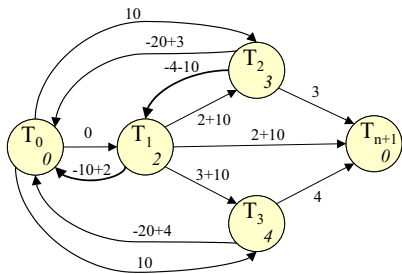
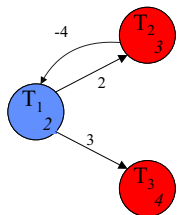
Temporal constraints $s_i + l_{ij} \leq s_j$ are transformed to:

$$\begin{aligned} s'_i - (a_i - 1) \cdot UB + l_{ij} &\leq s'_j - (a_j - 1) \cdot UB \\ s'_i + l_{ij} + (a_j - a_i) \cdot UB &\leq s'_j \end{aligned}$$

The transformed temporal constraint will look like $s'_i + l'_{ij} \leq s'_j$, where:

$$l'_{ij} = l_{ij} + (a_j - a_i) \cdot UB$$

Reduction from $PSm, 1 | \text{temp} | C_{max}$ to $PS1 | \text{temp} | C_{max}$



two dedicated resources
 T_1 on P^1 and T_2, T_3 on P^2

one resource

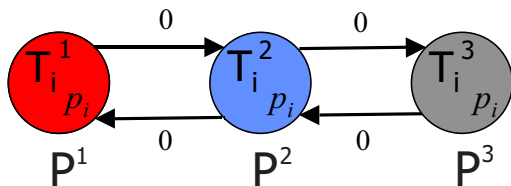
While minimizing the completion time of T_{n+1} , we push tasks T_1, T_2 and T_3 "to the left" due to the edges entering T_{n+1}

Multiprocessors Task

Transformation of multiprocessor task problem to $PSm, 1 | \text{temp} | C_{max}$

- create as many virtual tasks as there are processors needed to execute the physical tasks
- ensure that the virtual tasks of the given physical task start at the same time - this is done by two edges with weight $l_{ij} = l_{ji} = 0$. Consequently $s_i \leq s_j$ and $s_j \leq s_i$.

Example: Task T_i needs resources (P^1, P^2, P^3) .



Project Scheduling with Dedicated Resources of Different Capacity $PSm, R | \text{temp} | C_{max}$

Resource $k \in \{1, 2, \dots, m\}$ has a capacity of $R_k \in \mathbb{Z}^+ \cup \{\infty\}$ units.

Task i requires $r_{ik} \in \mathbb{Z}_0^+$ units of resource k , with $0 \leq r_{ik} \leq R_k$.

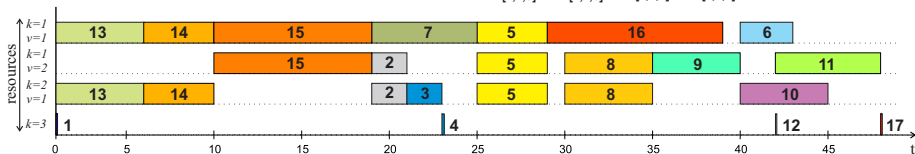
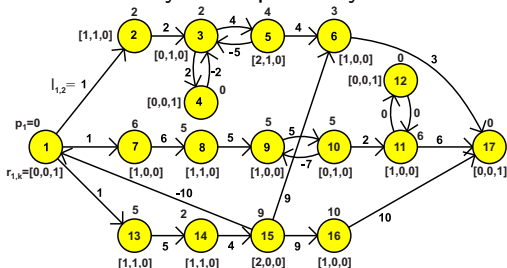
Multiprocessor tasks - Multiple resources may be required by one task.

Example:

$m = 3$

capacities of resources:

$R = (2, 1, \infty)$



Relative-order ILP Model for $PSm, R | \text{temp} | C_{max}$

The **assignment** $z_{ivk} \in \{0, 1\}$ is equal to 1 if task i is assigned to unit v of resource k , and 0 otherwise.

We define $\{i, j\} \in \mathcal{M}$ iff **task i and task j are assigned to resource k** of finite capacity (i.e., $\exists k \in \mathcal{R} : r_{ik} \cdot r_{jk} \geq 1$ and $R_k < \infty$) and therefore **we have to avoid a collision** of task i and task j . We define $\mathcal{V} = \{1, \dots, n\}$.

$$\min C_{max} \tag{1}$$

subject to:

$$s_j - s_i \geq l_{ij} \quad \forall (i, j) \in \mathcal{V}^2 : i \neq j \tag{2}$$

$$s_i - s_j + UB \cdot x_{ij} + UB \cdot y_{ij} \geq p_j \quad \forall (i, j) \in \mathcal{V}^2 : i \neq j, \{i, j\} \in \mathcal{M} \tag{3}$$

$$s_i - s_j + UB \cdot x_{ij} - UB \cdot y_{ij} \leq UB - p_i \quad \forall (i, j) \in \mathcal{V}^2 : i \neq j, \{i, j\} \in \mathcal{M} \tag{4}$$

$$-x_{ij} + y_{ij} \leq 0 \quad \forall (i, j) \in \mathcal{V}^2 : i \neq j, \{i, j\} \in \mathcal{M} \tag{5}$$




$$z_{ivk} + z_{jvk} - 1 \leq 1 - y_{ij} \quad \forall (i, j) \in \mathcal{V}^2, \forall k \in \{1 \dots M\}, \forall v \in \{1, \dots, R_k\} : \tag{6}$$

$$i \neq j, \{i, j\} \in \mathcal{M}$$

$$\sum_{v=1}^{R_k} z_{ivk} = r_{ik} \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{R} : r_{ik} \geq 1, R_k < \infty \tag{7}$$

$$s_i + p_i \leq C_{max} \quad \forall i \in \mathcal{V} \tag{8}$$

- Constraints (3), (4), (5) and binary variables x_{ij} and y_{ij}
 - ① When $x_{ij} = 0$ and $y_{ij} = 0$, constraints (3) and (4) reduce to $s_j + p_j \leq s_i$, i.e., **j is followed by i on the same unit.**
 - ② When $x_{ij} = 1$ and $y_{ij} = 0$, constraints (3) and (4) reduce to $s_i + p_i \leq s_j$, i.e., **i is followed by j on the same unit.**
 - ③ When $x_{ij} = 1$ and $y_{ij} = 1$, constraints (3) and (4) are eliminated in effect and the activities i and j do not share the same unit.
Combination $x_{ij} = 0$ and $y_{ij} = 1$ is not feasible due to constraint (5).
- Constraint (6) states that **when $y_{ij} = 1$ then the activities do not share the same unit v of resource k** since $z_{ivk} + z_{jvk} \leq 1$.
- Constraint (7) states that each **task i is assigned to the appropriate number of units r_{ik}** for each resource k .

-  J. Błażewicz, K. Ecker, G. Schmidt, and J. Węglarz.
Scheduling Computer and Manufacturing Processes.
Springer, second edition, 2001.
-  Klaus Neumann, Christoph Schwindt, and Jürgen Zimmermann.
Project Scheduling with Time Windows and Scarce Resources.
Springer, 2003.
-  Sigrid Knust Peter Brucker.
Complexity results for scheduling problems.
<http://www.ict.kth.se/courses/ID2204/index.html>.