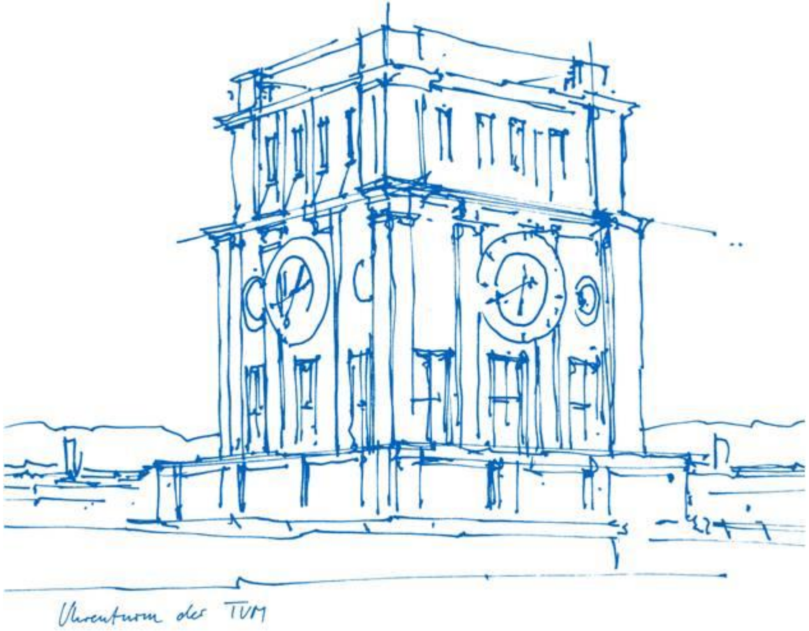
Complex Scheduling in Manufacturing and Services: Models, Methods, and Applications



Prof. Dr. Rainer Kolisch

TUM School of Management

Summer Term 2023

Course Organization

Lecture (Rainer Kolisch) / Exercise (Pia Ammann)

Lecture and exercise dates and locations:

- Lectures: Monday 09:45 am – 11:15 am in room 0540
- Exercises: Monday 11:30 am – 1:00 pm in room 0540

Course material:

- All material will be made available on Moodle until Sunday before the lecture.

Scope of the Course

Operations Research models and methods for scheduling problems in manufacturing and services.

Lecture

- Focus on models and methods.
- The topics will be presented based on textbooks and scientific publications.

Exercise

- Deepening the understanding of the content through exercises and assignments.
- Learning how to implement models and methods.
- Help in preparing the assignments (Python/Gurobi implementations with Google Colab / Jupyter Notebook).

Competences acquired in the Course

Competences:

- Models and methods for (complex) scheduling problems.
- Capability to model and solve practical scheduling problems.
- Knowledge in implementing (these) optimization models in Python/Gurobi

Where are these competences valuable:

- Analyzing, modelling optimizing, and planning (complex) scheduling problems
- Consulting

Required knowledge / modules:

- Management Science/Operations Research,
- Modelling, Optimization and Simulation,
- Introduction to Programming

Student Evaluation and Grading

Coding assignments:

- Student groups of size 2-3 have to undertake and hand in 3 assignments. The assignments consist of implementing optimization models and algorithms treated in class, using the Python/Gurobi API. In total, the assignments equal 50% of the grade. Assignments will be made available 2-3 weeks prior to the submission deadline.

Written examination:

- Open book exam of 90 minutes length. Questions will be similar to the content of the exercises. The exam equals 50% of the grade. Passing the exam is mandatory for completing the module successfully.

Lectures and Content

- Lecture 1: Modelling with minimum and maximum time lags
- Lecture 2: Calculating earliest and latest start and finish times
- Lecture 3: Time-based objective functions, regularity of objective functions
- Lecture 4: Weighted start time problem: LP and MCF-based formulation
- Lecture 5: RCPSP: Resource-based objective functions, MIP-formulations, MIP with pulse variables
- Lecture 6: RCPSP-models with objective function resource deviation and resource levelling
- Lecture 7: RCPSP-models with step variables and on-off variables
- Lecture 8: RCPSP-models with flow variables
- Lecture 9: Modeling production planning and scheduling problems with the RCPSP
- Lecture 10: Heuristics for solving the RCPSP: Serial and parallel schedule generation scheme, priority rule methods
- Lecture 11: Heuristics for solving the RCPSP: Improvement methods and metaheuristics
- Lecture 12: Heuristic for solving the resource leveling problem
- Lecture 13: Multi mode RCPSP: Models and metaheuristic
- Lecture 14: Stochastic and robust RCPSP

Literature

- Artigues et al. (2007): Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications, ISTE.
- Brucker und Knust (2012): Complex Scheduling, 2. Ed., Springer.
- Demeulemeester und Herroelen (2002): Project Scheduling – A Research Handbook, Kluwer.
- Neumann et al. (2003): Project Scheduling with Time Windows and Scarce Resources, 2. Ed., Springer.
- Schwindt, C. und Zimmermann, C. (2015): Handbook on Project Management and Scheduling Vol. 1-2, Springer.
- Zimmermann et al. (2010): Projektplanung – Modelle, Methoden, Management, 2. Ed, Springer.

References for scientific papers are given in the individual lectures.

Lecture 1: Modelling with Minimum and Maximum Time Lags

- Neumann et al. (2003): Project Scheduling with Time Windows and Scarce Resources, 2. Ed, Springer, Chapter 1.1. – 1.2 (pp. 1 – 9).
- Zimmermann et al. (2010): Projektplanung – Modelle, Methoden, Management, 2. Ed., Springer, pp. 43-119.

Graph Representation

We define a network N by means of a directed weighted graph $N = (V, E, \delta_{i,j})$ with

- set of nodes V,
- set of arcs E, and
- weight of arcs $\delta_{i,j} \in \mathbb{R}$.

Graph N is also called an activity-on-node (AoN) network because activities are modelled as nodes.

The set of nodes $V \coloneqq \{0,1, \dots, n+1\}$ represents a set of activities (jobs, tasks).

- Activity 0 is the unique start activity and activity $n+1$ is the unique finish activity.
- $p_i \geq 0$ is the deterministic duration of activity $i \in V$ with $p_0 = p_{n+1} = 0$.
- Let variable $S_i \geq 0$ be the start time of activity $i \in V$. We assume $S_0 = 0$.

The set of arcs E represents precedence relations between pair of activities. Arc $(i, j) \in E$ depicts a time difference $\delta_{i,j}$ between the start of activities $i, j \in V, i \neq j$.

Three Ways To Depict Activities with Time Constraints

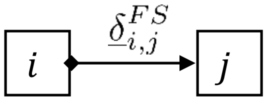
AON-Graph

Linear Program

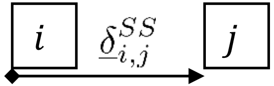
Gantt-Chart

Minimum Time Lags

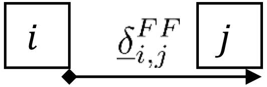
Finish-to-start: Activity j cannot start earlier than $\underline{\delta}_{i,j}^{FS}$ periods after the finish time of activity i.



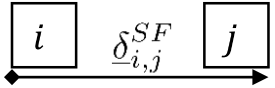
Start-to-start: Activity j cannot start earlier than $\underline{\delta}_{i,j}^{SS}$ periods after the start time of activity i.



Finish-to-finish: Activity j cannot finish earlier than $\underline{\delta}_{i,j}^{FF}$ periods after the finish time of activity i.



Start-to-finish: Activity j cannot finish earlier than $\underline{\delta}_{i,j}^{SF}$ periods after the start time of activity i.

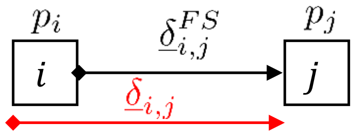


Note:

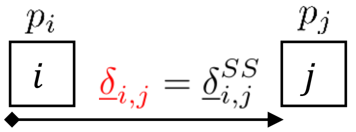
- All four type of minimum time lags have the domain $\underline{\delta}_{i,j} \in \mathbb{Z}$.
- Any type of minimum time lag can be transformed in any other type.

Transformation into Start-to-Start Time Lags

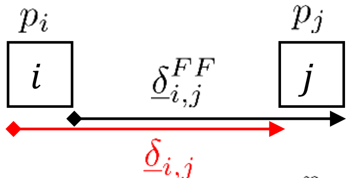
Finish-to-Start $\underline{\delta}_{i,j} = p_i + \underline{\delta}^{FS}_{i,j}$



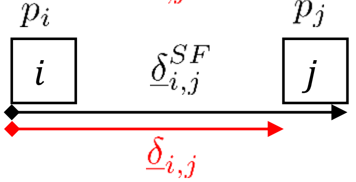
Start-to-Start $\underline{\delta}_{i,j} = \underline{\delta}^{SS}_{i,j}$



Finish-to-Finish $\underline{\delta}_{i,j} = p_i + \underline{\delta}^{FF}_{i,j} - p_j$



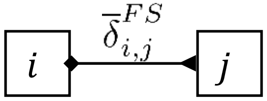
Start-to-Finish $\underline{\delta}_{i,j} = \underline{\delta}^{SF}_{i,j} - p_j$



Maximum Time Lags

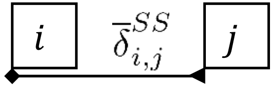
Finish-to-Start

Activity j must not start $\overline{\delta}^{FS}_{i,j}$ periods later than after the finish time of activity i.



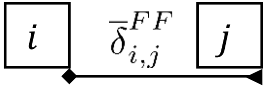
Start-to-Start

Activity j must not start $\overline{\delta}^{SS}_{i,j}$ periods later than after the start time of activity i.



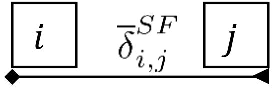
Finish-to-Finish

Activity j must not finish $\overline{\delta}^{FF}_{i,j}$ periods later than after the finish time of activity i.



Start-to-Finish

Activity j must note finish $\overline{\delta}^{SF}_{i,j}$ periods later than after the start time of activity i.



Note: Any type of maximum time lag can be transformed in any other type.

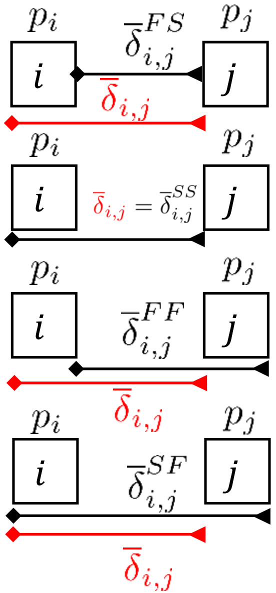
Transformation into Start-to-Start Time Lags

Finish-to-Start $\overline{\delta}_{i,j} = p_i + \overline{\delta}^{FS}_{i,j}$

Start-to-Start $\overline{\delta}_{i,j} = \overline{\delta}^{SS}_{i,j}$

Finish-to-Finish $\overline{\delta}_{i,j} = p_i + \overline{\delta}^{FF}_{i,j} - p_j$

Start-to-Finish $\overline{\delta}_{i,j} = \overline{\delta}^{SF}_{i,j} - p_j$


p_i
$\overline{\delta}^{FS}_{i,j}$
p_j
i
j
$\overline{\delta}_{i,j}$
p_i
p_j
i
$\overline{\delta}_{i,j} = \overline{\delta}^{SS}_{i,j}$
j
p_i
p_j
i
$\overline{\delta}^{FF}_{i,j}$
j
$\overline{\delta}_{i,j}$
p_i
p_j
i
$\overline{\delta}^{SF}_{i,j}$
j
$\overline{\delta}_{i,j}$

In the following we will consider only start-to-start precedence relations.

Examples Modelling using Minimum and Maximum Time Lags

Let $i, j \in V\{0, n+1\}$ be non-dummy activities with $i \neq j$ and $S_0 = 0$ be the start time of the dummy start activity 0.

- Earliest start time t_i (ready date, head) of i: $\underline{\delta}_{0,i} = t_i$

  earliest possible time at which activity i can start, assuming that all its preceding activities have been completed as early as possible.
- Minimum duration τ_i until the finish time of the project (tail): $\underline{\delta}_{i,n+1} = \tau_i$
- Deadline of i is t_i: $\overline{\delta}_{0,i} = t_i - p_i$
- Activity $\overset{j}{i}$ starts immediately after the finish time of activity $\overset{i}{j}$: $\underline{\delta}_{i,j} = \overline{\delta}_{i,j} = p_i$

  j is the direct successor of i
- Simultaneous start of activities i and j: $\underline{\delta}_{i,j} = \overline{\delta}_{i,j} = 0$
- Start of activity i at time t_i: $\underline{\delta}_{0,i} = \overline{\delta}_{0,i} = t_i$

  both activities start at the same time
- Start of activity i within time window $[t_i, t'_i]$: $\underline{\delta}_{0,i} = t_i,\ \overline{\delta}_{0,i} = t'_i$

  time window for start of activity: between minum time lag and maximum timelag

Transformation of Maximum into Minimum Time Lag

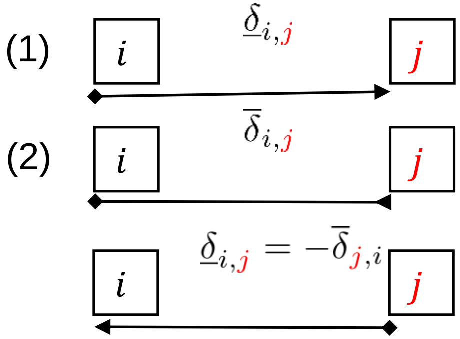
Let the variable $S_i \geq 0$ be the start time of activity $i \in V$

Minimum time lag written as linear constraint: $S_j - S_i \geq \underline{\delta}_{i,j} \quad (1)$

Maximum time lag written as linear constraint: $S_j - S_i \leq \overline{\delta}_{i,j} \quad (2)$

$$-1 \cdot (2) \Rightarrow S_i - S_j \geq -\overline{\delta}_{i,j} \quad (3)$$

(3) is a minimum time lag between j and i: $\Rightarrow S_i - S_j \geq \underline{\delta}_{j,i} \quad (4)$



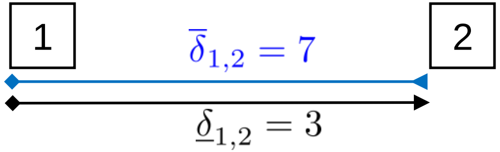
(1)
i
$\underline{\delta}_{i,j}$
j
(2)
i
$\overline{\delta}_{i,j}$
j
i
$\underline{\delta}_{i,j} = -\overline{\delta}_{j,i}$
j

<u>Observe</u>**:** A maximum time lag with positive (negative) value $\overline{\delta}_{i,j}$ between activities i and j equals a minimum time lag $\underline{\delta}_{j,i}$ with negative (positive) value between activities j and i, i.e., $\underline{\delta}_{i,j} = -\overline{\delta}_{j,i}$.

Example: Transforming Maximum into Minimum Time Lags

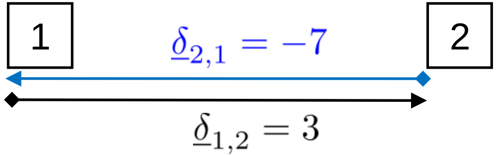
$\overline{\delta}_{1,2} = 7$: Activity 2 cannot start later than 7 periods after the start of activity 1.

$\underline{\delta}_{1,2} = 3$: Activity 2 cannot start earlier than 3 periods after the start of activity 1.



$\underline{\delta}_{2,1} = -7$: Activity 1 cannot start earlier than -7 periods after the start of activity 2.

$\underline{\delta}_{1,2} = 3$: Activity 2 cannot start earlier than 3 periods after the start of activity 1.

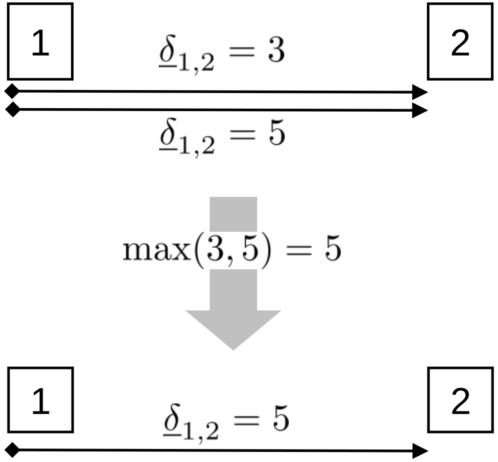

1
$\underline{\delta}_{2,1} = -7$
2
$\underline{\delta}_{1,2} = 3$

Reducing Multiple $\underline{\delta}_{i,j}$ for one (i,j)

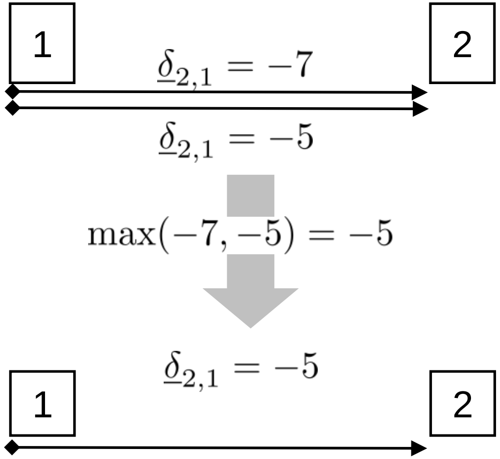
If there are multiple $\underline{\delta}_{i,j}$ for an activity pair (i,j) we select the one with maximum value and delete the others.

activity 2 can not start earlier 3 period after start of activity 1
So it is ok to start later but not earlier

Example:



1
2
$\underline{\delta}_{1,2} = 3$
$\underline{\delta}_{1,2} = 5$
$\max(3,5) = 5$
1
2
$\underline{\delta}_{1,2} = 5$



1
2
$\underline{\delta}_{2,1} = -7$
$\underline{\delta}_{2,1} = -5$
$\max(-7,-5) = -5$
$\underline{\delta}_{2,1} = -5$
1
2

Exercise 1

Consider the following information:

- Five activities $j = 1, \dots, 5$ with processing times 6, 4, 2, 4, and 2 have to be scheduled.
- Activities 1 and 3 can start immediately.
- Activity 5 has a release date of 8.
- Activity 2 cannot start before activity 1 has been processed for one period and has to be started 3 periods after the start of activity 1 at the latest.
- Activities 4 and 5 cannot be started before the completion of activity 3.
- Activity 4 has to be in process in the third period after the completion of activity 3.
- Activity 5 has to be in process in the second period after the completion of activity 4.
- Activity 3 has to be started at the latest one period after the start of the first activity.

Tasks:

1) Develop the AON-graph using the relevant of the 8 time lags.
2) Transform the AON-Graph so that only minimum time lags of the type start-to-start are used.
3) Formulate the problem as constraints of a linear program with continuous start time variables $S_i \geq 0$.
4) Solve the linear program, using an appropriate objective function of your choice.

Reference: Neumann, K., Schwindt, C. and Zimmermann, J. (2003): Project Scheduling with Time Windows and Scarce Resources, Springer, p. 8.

Lecture 2: Calculating Earliest and Latest Start and Finish Times

In the remainder of Part 1 we will only consider start-to-start minimum time lags by transforming all other time lags appropriately. Let denote $\delta_{i,j} = \underline{\delta}_{i,j}$.

- Neumann et al. (2003): Project Scheduling with Time Windows and Scarce Resources, 2. Ed, Springer, Chapter 1.1. – 1.2 (pp. 9 – 16).
- Zimmermann et al. (2010): Projektplanung – Modelle, Methoden, Management, 2. Ed., Springer, pp. 67-77.

Longest Path in the Network

- Let $N = (V, E, \delta_{i,j})$ be a network with activity set V and weights $\delta_{i,j}$ for all precedence relations $(i, j) \in E$ of the type start-to-start minimum time lags.

- The length of a path $\{0,1,2, \dots, r-1, r\}$ in N is $\sum_{k=1}^{r} \delta_{k-1,k}$.

- The longest path from node i to node j is denoted as $d_{i,j}$.

LP Time-Constrained Project Scheduling Problem

Let $S = (S_0, S_1, \dots, S_{n+1})$ be the vector of activity start time variables:

$$
\begin{array}{lll}
\text{Min } f(S) & & (1) \\
\text{s.t. } S_j - S_i \geq \delta_{ij} & \text{for all } (i,j) \in E & (2) \\
\quad S_i \geq 0 & \text{for all } i \in V & (3) \\
\quad S_0 = 0 & & (4) \\
\quad S_{n+1} \leq T^{max} & & (5)
\end{array}
$$

<u>Note</u>**:**

- T^{max} is the maximum duration allowed to process all activities. Instead of explicitly modeling (5), we could consider the maximum duration by constraint (2) as $S_0 - S_{n+1} \geq -\delta_{n+1,0}$ where $-\delta_{n+1,0} = T^{max}$ holds.

- $f(S)$ can be any function of the start times of the activities. The most common objective function is $f(S) = S_{n+1}$. Due to $S_0 = 0$,it minimizes the flow time (so-called makespan) of the project, i.e., the total time needed to process all activities.

Properties of Networks

Let $d_{i,j}$ denote the longest path from the start of activity i to the start of activity j.

For each network $N = (V, E, \delta_{i,j})$ we assume the following properties:

Property 1: No activity can be started before the start of the project, i.e. $d_{0,i} \geq 0$ holds for all $i \in V \backslash \{0\}$.

If Property 1 does not hold for an activity $i \in V \backslash \{0\}$, we introduce arc $(0, i)$ with $\delta_{0,i} = 0$.

Property 2: All activities have to be finished until the end of the project, i.e. $d_{i,n+1} \geq p_i$ holds for all $i \in V \backslash \{n+1\}$.

If Property 2 does not hold for an activity $i \in V \backslash \{n+1\}$ we introduce arc $(i, n+1)$ with $\delta_{i,n+1} = p_i$.

Algorithms for Computing Earliest and Latest Start Times

- Label-Correcting Algorithm (LCA)
- Floyd-Warshall Algorithm (FWA), also called Triple Algorithm

Label-Correcting Algorithm for Calculating Longest Paths

- Let $N = (V, E, \delta_{ij})$ be a network. The label correcting algorithm (LCA) calculates the longest paths between a single node $r \in V$ and all other nodes $i \in V \setminus \{r\}$.

```
Init:
    $d_{rr} := 0, p_{rr} := r$ and $Q :=< r]$;
    For all $j \in V \setminus \{r\}$:
        $d_{rj} := -\infty$ and $p_{rj} := -1$;

Main:
    While $Q \neq \emptyset$:
        Delete $i$ from the head of $Q$;
        For all $j \in Succ(i)$ with $d_{rj} < d_{ri} + \delta_{ij}$:
            $d_{rj} := d_{ri} + \delta_{ij},\ p_{rj} := i$;
                If $d_{rj} > (|V| - 1) \cdot \max\{\delta_{ij} | (i,j) \in E\}$ Then Stop “Positive Cycles”
                If $j \notin Q$ Then Insert $j$ at the end of $Q$;
```

Notation:

d_{rj}	Longest path from node r to node j
p_{rj}	Immediate predecessor of node j on the path from r to j
Q	FIFO – queue
$Succ(i)$	$:= \{j \in V | (i,j) \in E\}$

Comments on the LCA

1. Since LCA does not terminate if N contains cycles with positive length, a termination command is required.

   $(|V| - 1) \cdot \max\{\delta_{ij} | (i,j) \in E\}$ is an upper bound for d_{rj} for all $j \in V$. Hence,

   $$d_{rj} > (|V| - 1) \cdot \max\{\delta_{ij} | (i,j) \in E\}$$

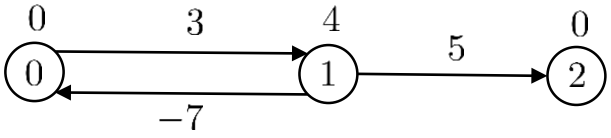
   indicates the existence of cycle of positive length which terminates LCA.

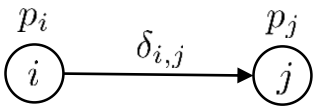
2. The run time complexity of LCA is $O(|V| \cdot |E|)$.

Example LCA without Positive Cycles

- Consider network $N = (V, E, \delta_{ij})$ with $r = 0$

Notation:

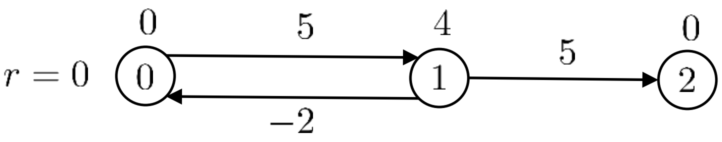




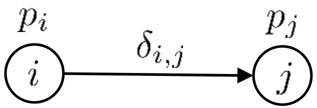
Iteration	Init		1		2		3	
i	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$
0	0	0	0	0	0	0	0	0
1	$-\infty$	-1	3	0	3	0	3	0
2	$-\infty$	-1	$-\infty$	-1	8	1	8	1
Q	$<0]$		$<1]$		$<2]$		$<]$	

$d_{0,0}$ is not updated since $d_{0,0} > d_{0,1} + \delta_{1,0}$

Example LCA with Positive Cycles



Notation:



Iteration	Init		1		2		3		4		5	
i	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$
0	0	0	0	0	3	1	3	1	3	1	6	1
1	$-\infty$	-1	5	0	5	0	5	0	8	0	8	0
2	$-\infty$	-1	$-\infty$	-1	10	1	10	1	10	1	13	1
Q	$< 0]$		$< 1]$		$< 2{,}0]$		$< 0]$		$< 1]$		$<]$	

$d_{0,0}$ is updated since $d_{0,0} < d_{0,1} + \delta_{1,0}$

$d_{0,2} = 13 > 10 \Rightarrow \text{Stop}$

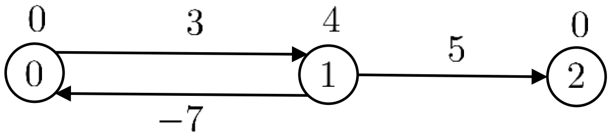
Calculation of Earliest Start and Finish Times with LCA

If the network does not have cycles of positive length, we can calculate for each activity $i \in V$ the earliest start time ES_i and the earliest finish time EF_i as follows:

$$ES_i \; = \; d_{0,i}$$
$$EF_i \; = d_{0,i} + \; p_i$$

<u>Example</u>:

$$ES_0 \underset{=}{} \; 0,, ES_1 \underset{=}{} \; 3, ES_2 \underset{=}{} \; 8$$
$$EF_0 \; = \; 0, EF_1 \; = \; 7, EF_2 \; = \; 8$$



Iteration	Init		1		2		3	
i	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$	$d_{0,i}$	$p_{0,i}$
0	0	0	0	0	0	0	0	0
1	$-\infty$	-1	3	0	3	0	3	0
2	$-\infty$	-1	$-\infty$	-1	8	1	8	1
Q	$<0]$		$<1]$		$<2]$		$<]$	

Calculation of Latest Start and Finish Times with the LCA

Let denote LS_i the latest start time of activity i and LF_i the latest finish time of activity i

Procedure:

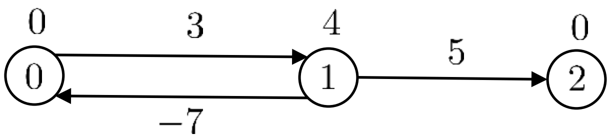
Step 1: Adding arc $(n+1{,}0)$ with $\delta_{n+1,0} = -T^{max}$ to $N = (V, E, \delta_{ij})$ gives $N' = (V, E', \delta_{ij})$ with $E' \coloneqq E \cup (n+1{,}0)$

Step 2: Reverse the arcs of $N' = (V, E', \delta_{ij})$ to obtain $N^H := (V, E^H, \delta_{ij}^H)$ with $(i, j) \in E' \Leftrightarrow (j, i) \in E^H$ and $\delta_{ij}^H =: \delta_{ij}$ for all $(j, i) \in E^H$.

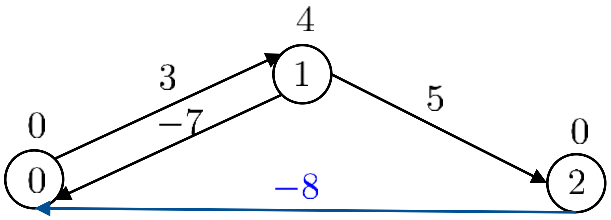
Step 3: Calculate $d_{0,i}^H$ for all $i \in V \backslash \{0\}$ by applying LCA to network N^H . If N^H does not contain positive cycles, we obtain $LS_i = -d_{0,i}^H$ and $LF_i = LS_i + p_i$ for all $i \in V \backslash \{0\}$. By definition we have $LS_0 {}_{=} LF_0 {}_{=} 0$.

Example Calculating LS- and LF-Times with the LCA

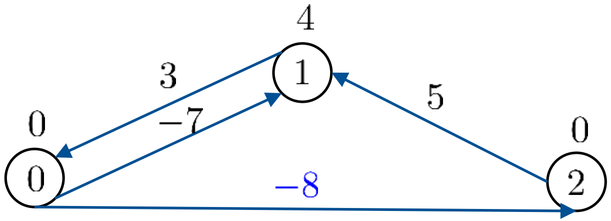
$N = (V, E, \delta_{ij})$



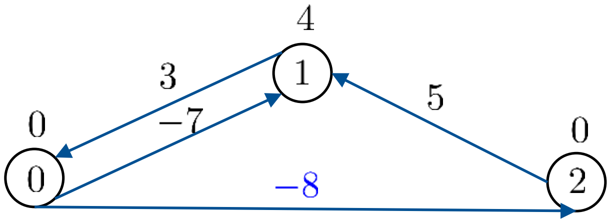
Step 1: $N' = (V, E', \delta_{ij})$



Step 2: $N^H := (V, E^H, \delta_{ij}^H)$



Example Calculating LS- and LF-Times with the LCA



Iteration	Init		1		2		3		4			
i	$d^H_{0,i}$	$p_{0,i}$	$d^H_{0,i}$	$p_{0,i}$	$d^H_{0,i}$	$p_{0,i}$	$d^H_{0,i}$	$p_{0,i}$	$d^H_{0,i}$	$p_{0,i}$	LS_i	LF_i
0	0	0	0	0	0	0	0	0	0	0	0	0
1	$-\infty$	-1	-7	0	-7	0	-3	2	-3	2	3	7
2	$-\infty$	-1	-8	0	-8	0	-8	0	-8	0	8	8
Q	$< 0]$		$< 1,2]$		$< 2]$		$< 1]$		$<]$			

$$LS_i = -d^H_{0,i}$$
$$LF_i = LS_i + p_i$$

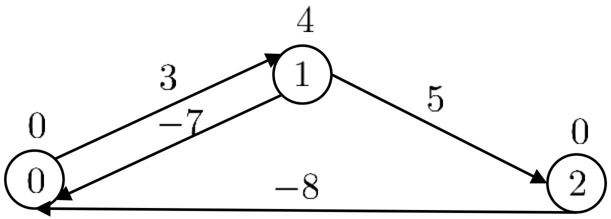
Calculating Earliest and Latest Start Times with the Floyd-Warshall-Algorithm

- Longest path calculations for all $(i,j) \in V \times V$ in network $N = (V, E, \delta_{ij})$.
- Cycle of positive length will be detected. No need for specific conditions.
- From the result we immediately obtain earliest and latest start times.

```
Init
  For all $i \in V$:
    $d_{i,i} \coloneqq 0, p_{i,i} \coloneqq i$;
    For all $j \in V \setminus \{i\}$:
      $d_{i,j} \coloneqq \delta_{i,j}$, if $(i,j) \in E$
              $-\infty$, otherwise;
      $p_{i,j} \coloneqq i$, if $d_{i,j} > -\infty$
             $-1$, otherwise;
Main
  For all $v \in V$:
    For all $i \in V \setminus \{v\}$ with $d_{i,v} > -\infty$:
      For all $j \in V \setminus \{v\}$ with $d_{i,j} < d_{i,v} + d_{v,j}$:
        $d_{i,j} \coloneqq d_{i,v} + d_{v,j}$ and $p_{i,j} \coloneqq p_{v,j}$.
```

For each pair (i,j) it is checked if the path from i to j traversing v is longer than the current path.

Example FW-Algorithm without Positive Cycles: Init

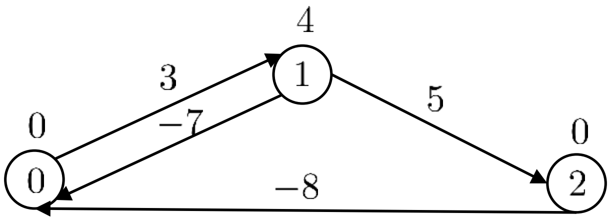


i \ j	0		1		2	
0	0	0	3	0	$-\infty$	-1
1	-7	1	0	1	5	1
2	-8	2	$-\infty$	-1	0	2

Notation:

$d_{i,j}$	$p_{i,j}$

Example FW-Algorithm without Positive Cycles: $v = 0$



Notation:

$d_{i,j}$ $p_{i,j}$

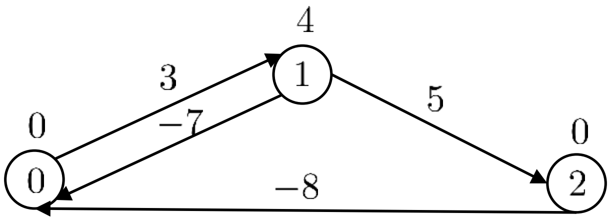
(d, p)-matrix before the iteration:

i \ j	0		1		2	
0	0	0	3	0	$-\infty$	-1
1	-7	1	0	1	5	1
2	-8	2	$-\infty$	-1	0	2

(d, p)-matrix after the iteration:

i \ j	0		1		2	
0	0	0	3	0	$-\infty$	-1
1	-7	1	0	1	5	1
2	-8	2	-5	0	0	2

Example FW-Algorithm without Positive Cycles: $v = 1$



Notation:

$d_{i,j}$	$p_{i,j}$

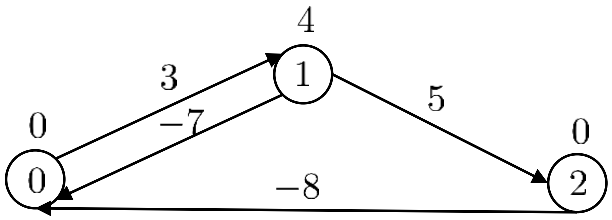
(d, p)-matrix before the iteration:

i \ j	0		1		2	
0	0	0	3	0	$-\infty$	-1
1	-7	1	0	1	5	1
2	-8	2	-5	0	0	2

(d, p)-matrix after the iteration:

i \ j	0		1		2	
0	0	0	3	0	8	1
1	-7	1	0	1	5	1
2	-8	2	-5	0	0	2

Example FW-Algorithm without Positive Cycles: $v = 2$



Notation:

$d_{i,j}$	$p_{i,j}$

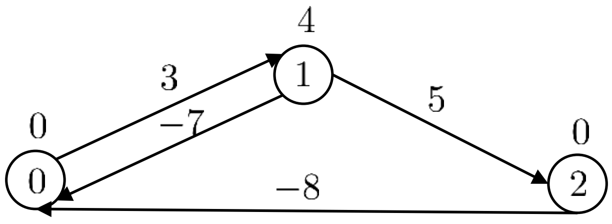
(d, p)-matrix before the iteration:

i \ j	0		1		2	
0	0	0	3	0	8	1
1	−7	1	0	1	5	1
2	−8	2	−5	0	0	2

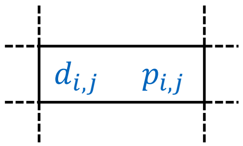
(d, p)-matrix after the iteration:

i \ j	0		1		2	
0	0	0	3	0	8	1
1	−3	2	0	1	5	1
2	−8	2	−5	0	0	2

Deriving Earliest and Latest Start Times



Notation:



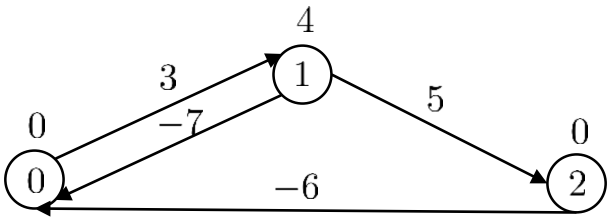
(d, p)-matrix at the end of the FW-algorithm:

$i \backslash j$	0		1		2	
0	0	0	3	0	8	1
1	−3	2	0	1	5	1
2	−8	2	−5	0	0	2

Earliest and latest start times:

$ES_i = d_{0,i}$	$LS_i = -d_{i,0}$
0	0
3	3
8	8

Example FW-Algorithm with positive cycles: Init

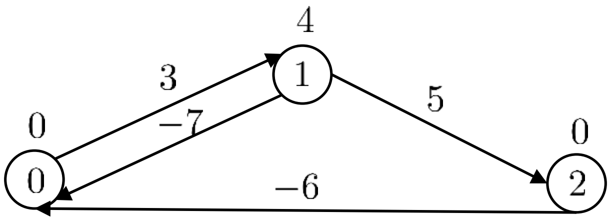


Notation:

$d_{i,j}$	$p_{i,j}$

$i \backslash j$	0		1		2	
0	0	0	3	0	$-\infty$	-1
1	-7	1	0	1	5	1
2	-6	2	$-\infty$	-1	0	2

Example FW-Algorithm with positive Cycles: $v = 0$



Notation:

$d_{i,j}$ $p_{i,j}$

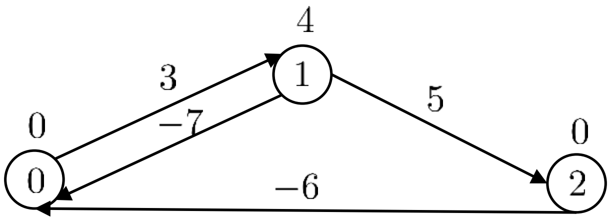
(d, p)-matrix before the iteration:

i \ j	0		1		2	
0	0	0	3	0	$-\infty$	-1
1	-7	1	0	1	5	1
2	-6	2	$-\infty$	-1	0	2

(d, p)-matrix after the iteration:

i \ j	0		1		2	
0	0	0	3	0	$-\infty$	-1
1	-7	1	0	1	5	1
2	-6	2	-3	0	0	2

Example FW-Algorithm with positive Cycles: $v = 1$



Notation:

$d_{i,j}$	$p_{i,j}$

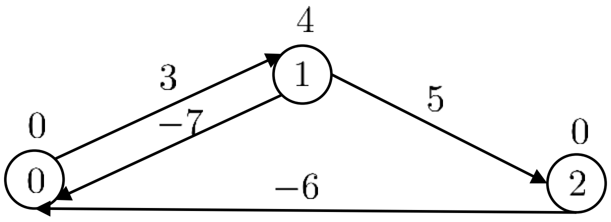
(d, p)-matrix before the iteration:

i \ j	0		1		2	
0	0	0	3	0	$-\infty$	-1
1	-7	1	0	1	5	1
2	-6	2	-3	0	0	2

(d, p)-matrix after the iteration:

i \ j	0		1		2	
0	0	0	3	0	8	1
1	-7	1	0	1	5	1
2	-6	2	-3	0	2	1

Example FW-Algorithm with positive Cycles: $v = 2$



Notation:

$d_{i,j}$	$p_{i,j}$

(d, p)-matrix before the iteration:

i \ j	0		1		2	
0	0	0	3	0	8	1
1	−7	1	0	1	5	1
2	−6	2	−3	0	2	1

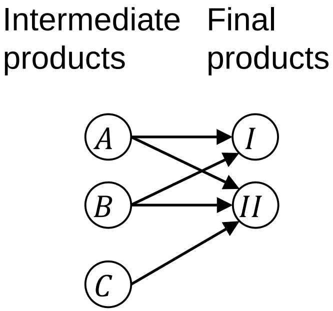
(d, p)-matrix after the iteration:

i \ j	0		1		2	
0	2	2	5	0	8	1
1	−1	2	2	0	5	1
2	−6	2	−3	0	2	1

In case of positive cycles, the FW-algorithm ends with $d_{i,i} > 0$ for at least one $i \in V$.

Application: Scheduling Make-to-Order Manufacturing

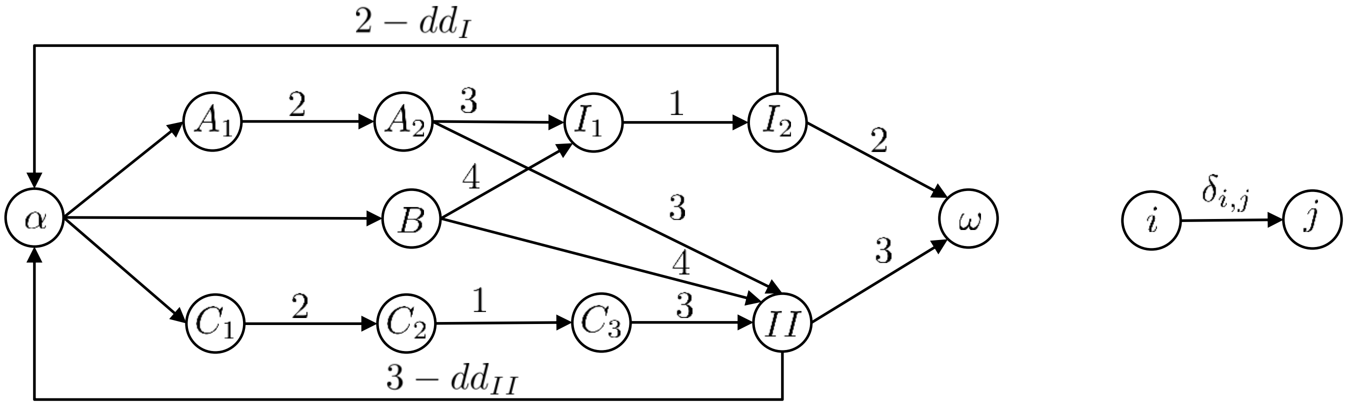
Example:



Process steps		Due dates
A_1: 2 periods A_2: 3 periods	I_1: 1 periods I_2: 2 periods	dd_I
B: 4 periods	II: 3 periods	dd_{II}
C_1: 2 periods C_2: 1 periods C_3: 3 periods		

See Example 2.43, pp. 202-203 in Zimmermann et al. (2010)

Application: Scheduling Make-to-Order Manufacturing



Exercise: Assuming due dates $dd_I = 10$ and $dd_{II} = 12$, calculate the earliest start times and the minimum project duration using the LC-algorithm or the FW-algorithm.

Exercise 2

Continuation of Exercise 1

1. Calculate the earliest start and finish times with the LCA.
2. Assume $LF_6 = EF_6$ and calculate the latest start and finish times with the LCA.
3. Calculate the latest start and finish times with the LCA assuming a latest project finish time of $LF_6 = 14$.
4. Assume $LF_6 = 14$ and calculate the earliest and latest start times with the FWA.

Lecture 3:
Time-based Objective Functions and the Weighted Start Time Problem

Time-based Objective Functions:

- Min makespan (M): $f(S) = S_{n+1}$
- Min mean flow time (MFT): $\frac{1}{\mathrm{n+2}} \sum_{i \in V} (S_i + p_i)$
- Min sum weighted start times (WST): $f(S) = \sum_{i \in V} w_i \cdot S_i$ with $w_i \in \mathbb{R}$ for all $i \in V$

Remark**:** Since objective functions M, MFT and WST are linear in S, the time-constrained project scheduling problem (2) – (5), see slide 25, can be solved in polynomial time.

Time-based Objective Functions

- Min sum earliness and tardiness (E+T): Typical in order management

How much we are ahead of plan time

finish time

We only take the value if the value is positive

We finish before the due date

$$f(S) = \sum_{i \in V} c_i^E \cdot (d_i - S_i - p_i)^+ + c_i^T \cdot (S_i + p_i - d_i)^+$$

time we would like to finish

Late and early

with d_i = due date of activity $i \in V$

c_i^E = earliness cost of activity i cost of machine sit there

c_i^T =tardiness cost of activity i cost of penalty from customer

$x^+ = \max\{x, 0\}$ for $x \in \mathbb{R}$

Remark**:** Since objective function E+T is linear in S, the time-constrained project scheduling problem (2) – (5) can be solved in polynomial time.

Linearization of ET-objective function

Regularity of Objective Functions

Definition:

Function $f\colon \mathbb{R}^{n+2} \rightarrow \mathbb{R}$ is regular if it is non-decreasing in the start time S_i of activities. That is, for two start time vectors S and S' with $S_i \leq S_i^{'}$ for all $i \in V$ it holds $f(S) \leq f(S')$.

$f(S)$	regular
M	yes
MFT	yes
WST with $w_i \geq 0$	yes
WST with $w_i \in \mathbb{R}$	no
E+T	no

When it is not regular, it is easier for heuristics to find the optimal solution
We just need to try out the schedule as soon as possible
See more photos

Min Weighted Start Time Problem (WSTP)

Min $\sum_{i \in V} w_i \cdot S_i$ (1)

s.t.

$S_j - S_i \geq \delta_{i,j}$ for all $(i,j) \in E$ (2)

$S_i \geq 0$ for all $i \in V$ (3)

$S_0 = 0$ (4)

$S_{n+1} \leq d_{0,n+1}$ (5)

WSTP is a linear program and can be solved with the Simplex Algorithm.

However, we can transform WSTP in a min cost flow (MCF) problem which can be solved with specific efficient algorithms such as the cycle-canceling algorithm.

Due to this, we don’t have to apply the Simplex and are more efficient than using the Simplex.

Recap: Rules for obtaining the Dual Problem

Maximization	↔	Minimization
Constraint $\leq$		Variable ≥ 0
Constraint $\geq$		Variable ≤ 0
Constraint $=$		Variable $\in \mathbb{R}$
Variable ≥ 0		Constraint $\geq$
Variable ≤ 0		Constraint $\leq$
Variable $\in \mathbb{R}$		Constraint $=$

Reformulation of the Primal Program: Streamlining the WSTP

Min $\sum_{i \in V} w_i \cdot S_i$ (1)

s.t. $S_j - S_i \geq \delta_{i,j}$ for all $(i,j) \in E$ (2)

$S_0 = 0$ (3)

$S_{n+1} \leq d_{0,n+1}$ (4) ⇒ holds due to (2)

$S_i \geq 0$ for all $i \in V$ (5) ⇒ holds due to (2) and (3) (*

Min $\sum_{i \in V} w_i \cdot S_i$ (1)

s.t. $S_j - S_i \geq \delta_{i,j}$ for all $(i,j) \in E$ (2)

$S_0 = 0$ (3)

$S_i \in \mathbb{R}$ for all $i \in V$ (4’)

(* See slide 23:
Property 1: No activity can be started before the start of the project, i.e., $d_{0\ i} \geq 0$ holds for all $i \in V \backslash \{0\}$.
If the property does not hold for an activity $i \in V \backslash \{0\}$ we introduce arc $(0, i)$ with $\delta_{0,i} = 0$.

Obtaining the Dual of the WSTP

Primal WSTP:				Dual Variables and Constraints
Min	$\sum_{i \in V} w_i \cdot S_i$		(1)	
s.t.	$S_j - S_i \geq \delta_{i,j}$	for all $(i,j) \in E$	(2)	$x_{ij} \geq 0$ for all $(i,j) \in E$
	$S_0 = 0$		(3)	$z \in \mathbb{R}$
	$S_i \in \mathbb{R}$	for all $i \in V$	(4’)	Equation of type “=“

Dual WSTP:

Max	$\sum_{(i,j)} \delta_{ij} \cdot x_{ij} + 0 \cdot z$		(1)
s.t.	$\sum_{h \in Pred(0)} x_{h,0} - \sum_{j \in Succ(0)} x_{0,j} + z = w_0$	for $i = 0$	(2)
	$\sum_{(h,i) \in E} x_{h,i} - \sum_{(i,j) \in E} x_{i,j} = w_i$	for all $i \in V \backslash \{0\}$	(3)
	$x_{ij} \geq 0$	for all $(i,j) \in E$	(4)
	$z \in \mathbb{R}$		(5)

with $Pred(j) := \{i \in V \mid (i,j) \in E\}$ and $Succ(i) := \{j \in V \mid (i,j) \in E\}$.

Streamlining the Dual WSTP

Max $\sum_{(i,j)} \delta_{ij} \cdot x_{ij} + 0 \cdot z$ (1)

s.t. $\sum_{h \in Pred(0)} x_{h,0} - \sum_{j \in Succ(0)} x_{0,j} + z = w_0$ for $i = 0$ (2)

$\sum_{(h,i) \in E} x_{h,i} - \sum_{(i,j) \in E} x_{i,j} = w_i$ for all $i \in V \setminus \{0\}$ (3)

$x_{ij} \geq 0$ for all $(i,j) \in E$ (4)

$z \in \mathbb{R}$ (5)

- Constraints (2) for $i = 0$ and (3) for $i \in V \setminus \{0\}$ have the same structure. The only difference is variable z.
- However, since z has coefficient 0 in the objective function we can set it to any value in $\mathbb{R}$.
- Choosing $z := \sum_{i \in V \setminus \{0\}} w_i$, moving z to the RHS of (2) and remembering that $w_0 = 0$ we obtain $-\sum_{i \in V \setminus \{0\}} w_i$ as new RHS of (2).

Max $\sum_{(i,j)} \delta_{ij} \cdot x_{ij}$ (1)

s.t. $\sum_{(h,i) \in E} x_{h,i} - \sum_{(i,j) \in E} x_{i,j} = w_i$ for all $i \in V$ (2)

$x_{ij} \geq 0$ for all $(i,j) \in E$ (3)

with $w_o = -\sum_{i \in V \setminus \{0\}} w_i$

idea to start with infeasible solution and then find the one that is feasibale

General Min Cost Flow Problem (MCFP)

Let $N = (V, E, l_{ij}, u_{ij})$ be a flow network with node set V, arc set E, for each arc $(i, j) \in E$ lower bound of the flow l_{ij} , upper bound of the flow u_{ij}, and variable flow cost c_{ij} as well as for each node $i \in V$ demand $b_i \in \mathbb{R}$.

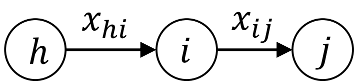
Min $\sum_{(i,j)\in E} c_{ij} \cdot x_{ij}$ (1)

s.t.

$\sum_{(h,i)\in E} x_{hi} - \sum_{(i,j)\in E} x_{ij} = b_i$ for all $i \in V$ (2)

$l_{ij} \le x_{ij} \le u_{ij}$ for all $(i, j) \in E$ (3)

$x_{ij} \ge 0$ for all $(i, j) \in E$ (4)



with $\sum_{i\in V} b_i = 0$ (⇒ total supply = total demand in the network)

Note: $b_i > 0$ ⇒ demand note, $b_i < 0$ supply node, and $b_i = 0$ ⇒ transshipment node.

The MCFP can be solved as a linear program with the Simplex algorithm. However, there are more efficient solution procedures available, which are not using the Simplex. Examples are the Cycle Canceling algorithm or the Out-of-Kilter algorithm.

Casting the Dual of the WSTP as MCFP

Negative cycle: sume of the value on each arces on the cycle
Meaning: When sending one unit on the cyle the cost will reduced by the value
Example in the foto

instead of sending the unit via 2 node, 2 arces, we send it directly
then we save by value of the acres minus the value of direct arce

next question is to detact the negative cycle

MCFP

Min $\sum_{(i,j)\in E} c_{ij} \cdot x_{ij}$ (1)

s.t. $\sum_{(h,i)\in E} x_{h,i} - \sum_{(i,j)\in E} x_{i,j} = b_i$ for all $i \in V$ (2)

$l_{ij} \leq x_{ij} \leq u_{ij}$ for all $(i,j) \in E$ (3)

$x_{ij} \geq 0$ for all $(i,j) \in E$ (4)

D-WSTP

Max $\sum_{(i,j)\in E} \delta_{ij} \cdot x_{ij}$ (1)

s.t. $\sum_{(h,i)\in E} x_{h,i} - \sum_{(i,j)\in E} x_{i,j} = w_i$ for all $i \in V$ (2)

$x_{ij} \geq 0$ for all $(i,j) \in E$ (3)

with $w_o = -\sum_{i\in V\setminus\{0\}} w_i$

D-WSTP as MCFP

Min $\sum_{(i,j)\in E} -\delta_{ij} \cdot x_{ij}$ (1)

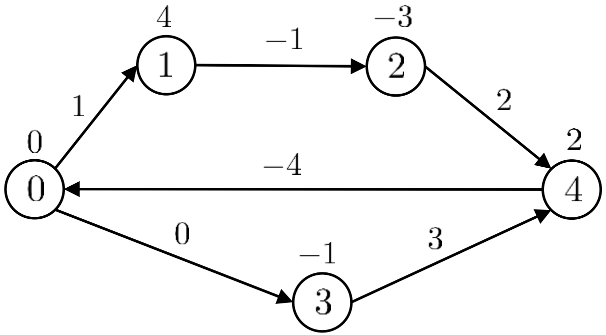
s.t. $\sum_{(h,i)\in E} x_{h,i} - \sum_{(i,j)\in E} x_{i,j} = w_i$ for all $i \in V$ (2)

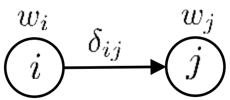
$0 \leq x_{ij} \leq \infty$ for all $(i,j) \in E$ (3)

with $w_o = -\sum_{i\in V\setminus\{0\}} w_i$

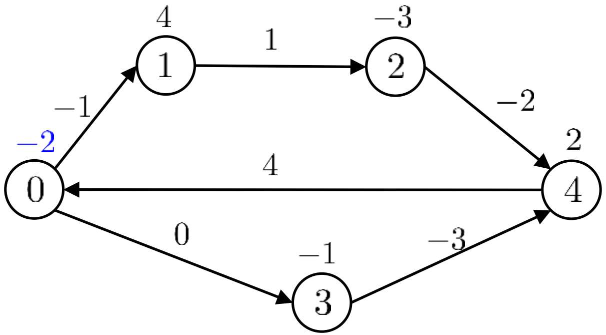
Example for Modelling the WSTP as MCFP

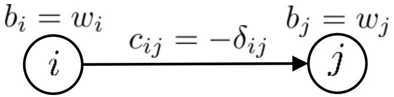
WSTP





D-WSTP as MCFP

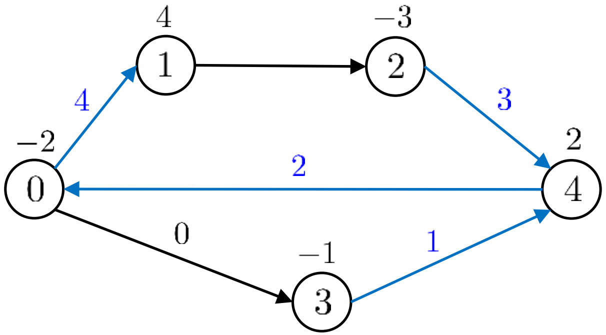


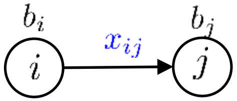


$$w_o = -\sum_{i \in V \setminus \{0\}} w_i$$

Transforming the MCFP-Solution into the WSTP-Solution

Optimal MCFP-Solution

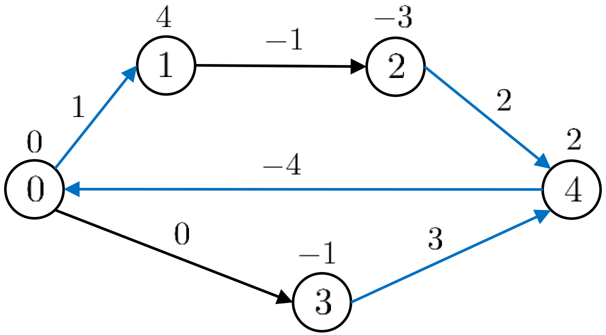


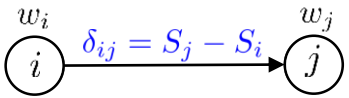


From the strong duality theorem we know, that for a variable $x > 0$ in the dual problem, “=“ holds in the associated constraint of the primal problem.

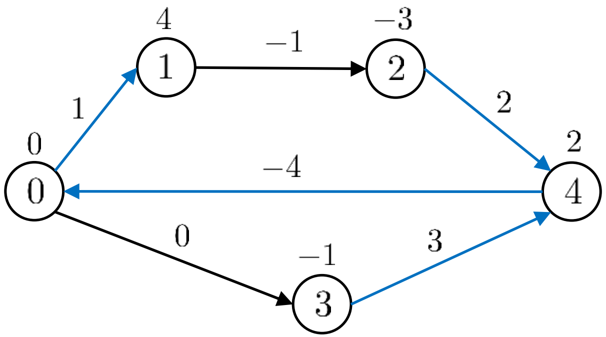
Applying the strong duality theorem we obtain for each $x_{i,j} > 0$ in the dual MCFP-problem $S_j - S_i = \delta_{i,j}$ in the primal WST-problem.

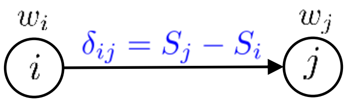
WST-Problem with binding constraints





Example for Transforming the MCFP- into the WSTP-Solution





We obtain the following binding constraints:

$$S_1 - S_0 = 1$$
$$S_0 - S_4 = -4$$
$$S_4 - S_2 = 2$$
$$S_4 - S_3 = 3$$

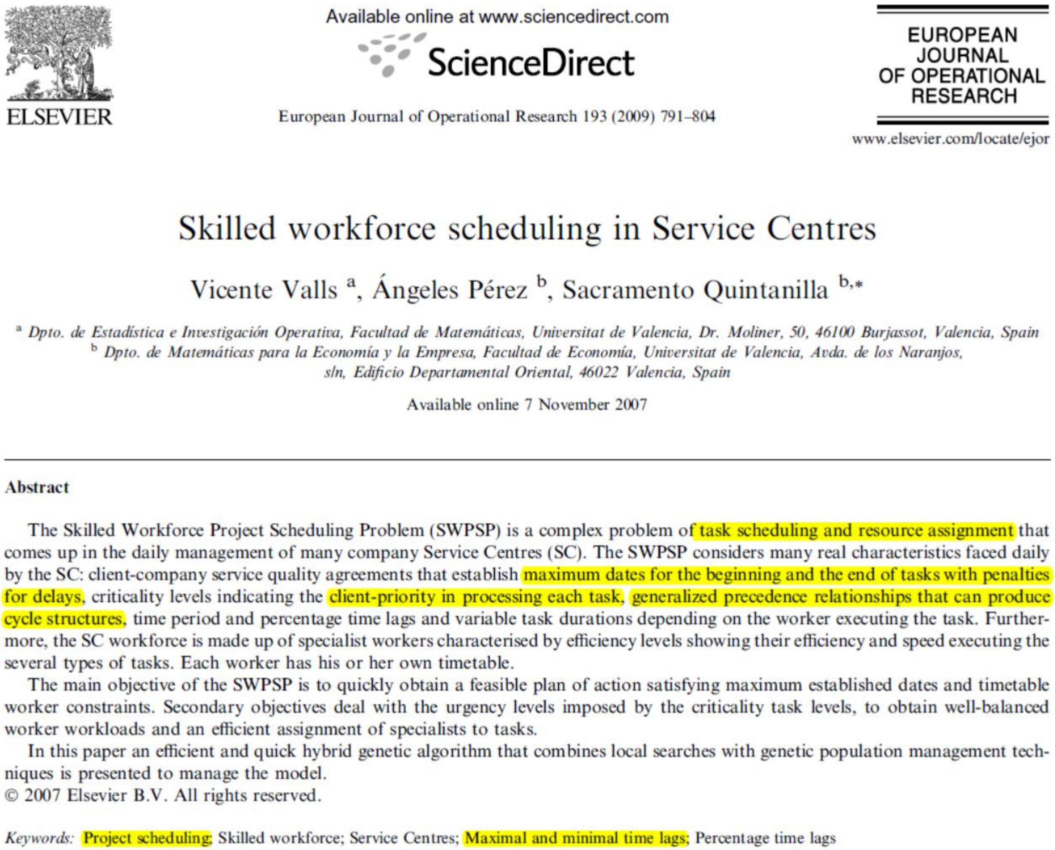
With $S_0 = 0$ we can derive:

$$S_0 := 0$$
$$S_1 = 1$$
$$S_4 = 4$$
$$S_2 = 2$$
$$S_3 = 1$$

Application: Workforce Scheduling in Service Centers

Reference: Valls, V., Perez, A., Quintanilla, S. (2009): Skilled workforce scheduling in service centers, European Journal of Operational Research, Vol. 193, pp. 791-804.

Vicente et al. (2009): Workforce Scheduling



ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 193 (2009) 791–804

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Skilled workforce scheduling in Service Centres

Vicente Valls [a], Ángeles Pérez [b], Sacramento Quintanilla [b,*]

[a] *Dpto. de Estadística e Investigación Operativa, Facultad de Matemáticas, Universitat de Valencia, Dr. Moliner, 50, 46100 Burjassot, Valencia, Spain*
[b] *Dpto. de Matemáticas para la Economía y la Empresa, Facultad de Economía, Universitat de Valencia, Avda. de los Naranjos, s/n, Edificio Departamental Oriental, 46022 Valencia, Spain*

Available online 7 November 2007

Abstract

The Skilled Workforce Project Scheduling Problem (SWPSP) is a complex problem of task scheduling and resource assignment that comes up in the daily management of many company Service Centres (SC). The SWPSP considers many real characteristics faced daily by the SC: client-company service quality agreements that establish maximum dates for the beginning and the end of tasks with penalties for delays, criticality levels indicating the client-priority in processing each task, generalized precedence relationships that can produce cycle structures, time period and percentage time lags and variable task durations depending on the worker executing the task. Furthermore, the SC workforce is made up of specialist workers characterised by efficiency levels showing their efficiency and speed executing the several types of tasks. Each worker has his or her own timetable.

The main objective of the SWPSP is to quickly obtain a feasible plan of action satisfying maximum established dates and timetable worker constraints. Secondary objectives deal with the urgency levels imposed by the criticality task levels, to obtain well-balanced worker workloads and an efficient assignment of specialists to tasks.

In this paper an efficient and quick hybrid genetic algorithm that combines local searches with genetic population management techniques is presented to manage the model.

Keywords: Project scheduling; Skilled workforce; Service Centres; Maximal and minimal time lags; Percentage time lags

Application: Workforce Scheduling in Service Centers

email received: earliest time to process, have to be process in a given time
Based on customer prioritizeation has different deadlinet to answer

Characteristics of the problem:

take training to know how to use tool, email from customers, has to be answer

- Internal and external demand for service workers are depicted as activities
- General precedence constraints between activities
- Time windows for performing activities
- Weights for activities (higher weight → higher priority)
- The objective is to minimize the sum of the weighted start times.

high priority

Not treated here but considered in the paper:

- Resource constraints

NP hard problem

- Additional objective functions

Managerial results:

- Genetic algorithm for solving the model is intended to be implemented in a Service Center. Management Tools for task scheduling and the management of human resources.

Application: Workforce Scheduling in Service Centers

Characteristics of the problem:

- Internal and external demand for service workers are depicted as activities
- General precedence constraints between activities
- Time windows for performing activities
- Weights for activities (higher weight → higher priority)
- The objective is to minimize the sum of the weighted start times.

Not treated here but considered in the paper:

- Resource constraints
- Additional objective functions

Managerial results:

- Genetic algorithm for solving the model is intended to be implemented in a Service Center. Management Tools for task scheduling and the management of human resources.

Scheduling in Service Centers: Notation

- Activity duration: d_j
- Latest start time of activity: ms_j

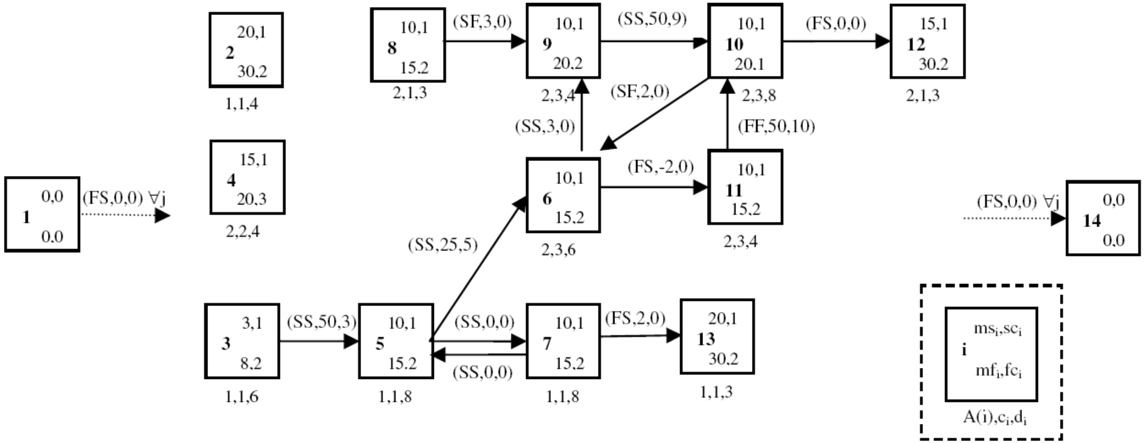
maximum start and finish, time window for the activity

- Latest finish time of activity: mf_j
- Per period cost for delayed activity start: sc_j
- Per period cost for delayed activity finish: fc_j
- Criticality of activity: c_j
- General precedence constraints: $d_{ij} = (GPR, p, \alpha)$

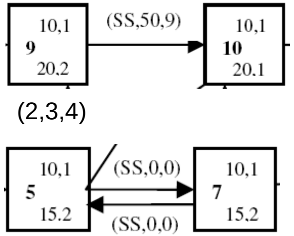
general precedence relationship

  - $GPR = \{SS, FF, SF, FS\}$,
  - p=time lag (0=absolute; > 0 percent of the duration of activity α),
  - α = in case of $p > 0$, activity of which at least p% of the duration has to be undertaken before the successor can start.
- $A(j)$: skilled required for processing activity j
- $PDUR$: Planning horizon

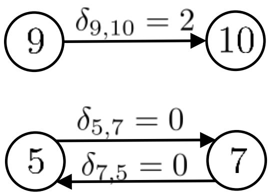
Scheduling in Service Centers: Activity Network



Notation Valls et al.(2009):



Notation in this course:



Scheduling in Service Centers: Model

$$\text{Min} \sum_{j=1}^{n} c_j \cdot s_j \qquad (1)$$

subject to

$$s_j - s_i \geqslant \mathrm{SS}_{ij} \quad \forall (i,j) \in E_{\mathrm{SS}}, \qquad (2)$$

$$f_j - s_i \geqslant \mathrm{SF}_{ij} \quad \forall (i,j) \in E_{\mathrm{SF}}, \qquad (3)$$

$$s_j - f_i \geqslant \mathrm{FS}_{ij} \quad \forall (i,j) \in E_{\mathrm{FS}}, \qquad (4)$$

$$f_j - f_i \geqslant \mathrm{FF}_{ij} \quad \forall (i,j) \in E_{\mathrm{FF}}, \qquad (5)$$

$$s_i \leqslant \mathrm{ms}_i \quad \forall i \in V, \qquad (6)$$

start time has to be before or at the maximum start time

$$f_i \leqslant \mathrm{mf}_i \quad \forall i \in V, \qquad (7)$$

$$0 \leqslant s_i \leqslant \mathrm{PDUR} \quad \forall i \in V, \qquad (10)$$

$$s_1 = 0, \qquad (11)$$

$$f_i = s_i + d_i \qquad \forall i \in V \qquad (12)$$

$$f_i, s_i \geq 0 \qquad \forall i \in V \qquad (13)$$

Stochastic problem
every time the request comming, we can not know, but we can forecast it
we solved the problem for entire planning horizon
constant planing next 5 minutes again
New planning with the request we have

other solution: using the online planning, replan any time new request coming iin

How often do we planning?

Lecture 4: Scheduling with Time and Resource Constraints

- Neumann et al. (2003): Project Scheduling with Time Windows and Scarce Resources, 2. Ed, Springer, Chapter 1.1. – 1.2 (pp. 9 – 16).
- Zimmermann et al. (2010): Projektplanung – Modelle, Methoden, Management, 2. Ed., Springer, pp. 67-77.

Renewable Resources

resource available in every period again: Ex worker work 8 hours a day, one day next day available again (worker, machine, room,
Money is not renewable, it always gone from the budget

- $\mathcal{R}$ is a set of renewable resources.
- Each resource $k \in \mathcal{R}$ has a capacity of $R_k \in \mathbb{Z}_0^+$ at any time.
- Activity $i \in V$ requires $r_{ik} \in \mathbb{Z}_0^+$ capacity units of resource k at any time while being processed.

schedule is the vector of n+2 startime,

- For schedule S ∈ $\mathbb{R}^{n+2}$ the demand for resource k at time t is $r_k(S,t)$.

For ex: Resource {1,2} we have 2 workers R2
1 room R1
imagine we have 3 activities
r[ik] for
Activity 1: r11 = 1, r12 = 1, p1 = 3
Activity 2: r21 = 0, r22 = 2, p2 = 5

foto for more details

in each resource we have the contraints and can
not use more than the constrain
For capacity of R1 is 1, R2 is 2

If we dont have resource, the optimal schedule will look like this
optimal schedule for resource unconstraint problem
minimize make span
Schedule here will be S(0, 0, 0, 5) 1 start at 0l, 2 start at 0, 3 start at 5
S (0, S1, S2, S3)
foto

Conclusion: resource demand profile is the green line R1(s,t), r2(s,t)
resource depend on the startime and the time
after the schedule we can derive the resource profile

Activities in Process at time t

A(S,0) = {1,2}
A(S,3) = {2}
A(s,5) = {}

For schedule S ∈ $\mathbb{R}^{n+2}$ the set of activities which are in process at time t is

for the time t we look at, the start time has to start now or before,

$$A(S,t) := \{i \in V \mid S_i \le t < S_i + p_i\}$$

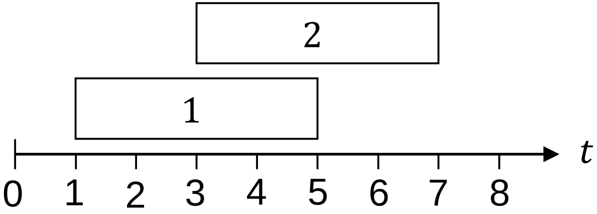
activities that active at one given point in time

start time plus processing time

start time always be the point in time not preriod shceulde

Example:

$S = (1,3),\ p = (4,4)$



t	0	1	2	3	4	5	6	7	8
$A(S,t)$	$\emptyset$	$\{1\}$	$\{1\}$	$\{1,2\}$	$\{1,2\}$	$\{2\}$	$\{2\}$	$\emptyset$	$\emptyset$

2 just start here, both active

1 already finish

cutting time then see which activities is there
Assuming all the time is integer

Total Resource Demand at time t: $r_k(S,t)$

S here come from the solution of the schedule

For resource $k \in R$ the total resource demand at time t is: $r_k(S,t) = \sum_{i \in A(S,t)} r_{i,k}$

summing up the resource demand over time

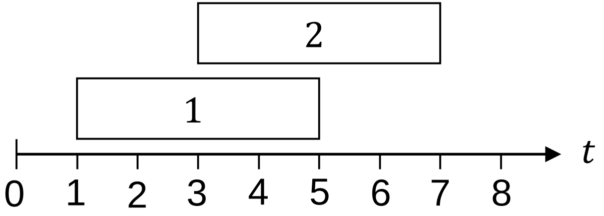
Example:

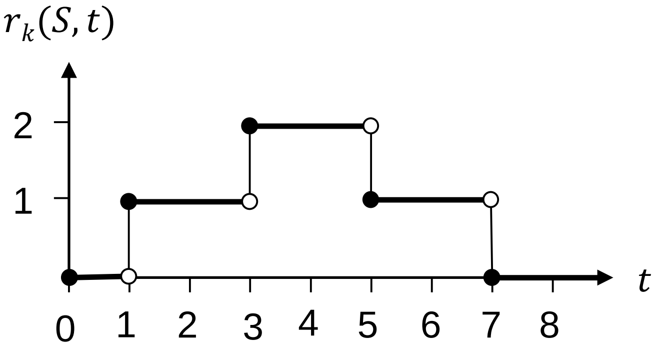
$$S = (1,3),\ p = (4,4)$$
$$R = \{1\}, r_{i,1} = (1,1)$$

A(S,0) = {1,2}
A(S,3) = {2}
A(s,5) = {}
r2(S,t) = r12 + r22 =3

in t = 0 have 2 active activity
in t = 3 have only 1 active
r22 = 2
Then go back to 0 when all activities finished

Conclusion: resource constraint rk(s,t) <=Rk with t = 0,...Tmax - 1





Renewable Resources: Application

Maximum number of workers which are allowed to work simultaneously due to space limitations.

Example: Maintenance of helicopters.

R_k = maximum number of workers which can work at the same time due to space limitations.

r_{ik} = 1 for each activity i which is undertaken by a worker.

Reference: Brimberg, J., Hurley, W.J. and Wright. R.E.: Scheduling workers in a constricted area. Naval Research Logistics, 43:143–149, 1996.

Modelling with Renewable Resources

- Maximum number of activities, which can be undertaken in parallel due to space restrictions.
- Incompatibility of two activities: The activities are not allowed to be processed in parallel / to overlap.
- Activities require resources to be processed, with limited number of resources available. Examples: Machines, lecture rooms, sport fields at sport tournaments, ….

fuel and electric can cause explosion can not do at the same time

Resource for the helicopter:
r
1 mechanics,
2 electrician,
3 information system (computer science)
4 work place (all the tool and equipment)
5 cockpit space

if you are maintaining the large plan, can not do all the activity due to constraint space problem

there some activities need to be done at the unparraled time for example Resource only have constraint of 1, 2 activities consume 1 each
Time tabling is a scheduling problem

Resource-related Objective Functions: Resource Investment

Resource Investment Problem (**RI**): Min total weighted peak resource demand

$$\text{Min } f(S) = \sum_{k \in R} c_k^I \max_{t=0}^{T^{\max}} r_k(S,t)$$

resource skyline, take the maximum of the whole planning horizon
Minimize the peak resource demand (schedule activity after each other

with c_k^I investment costs for one unit of resource $k \in R$.

Note: RI is non-linear but can be linearized by introducing a continuous variable.

Resource-related Objective Functions: Resource Deviation

Resource Deviation Problem (**RD**): Min total weighted positive deviation from given resource levels

$$\text{Minimize } f(S) = \sum_{k \in R} c_k^D \sum_{t=0}^{T^{\max}} \left(r_k(S,t) - Y_k\right)^+$$

sum up the several resource

given level of the resource. Determine where the resource use peak out of the threshold
Give the amount of work give to external outsourcing
Overcapacity problem

with Y_k level of resource $k \in R$

c_k^D per period and unit cost for exceeding resource level Y_k of resource $k \in R$.

Remarks:

- The objective function is non-linear, but can be linearized by introducing a continuous variable.
- Only positive deviations (demand > resource level) are modeled here. This problem is also termed resource overload problem.
- However, negative deviations such as opportunity cost for not used capacity can also be modeled.

Resource-related Objective Functions: Resource Levelling

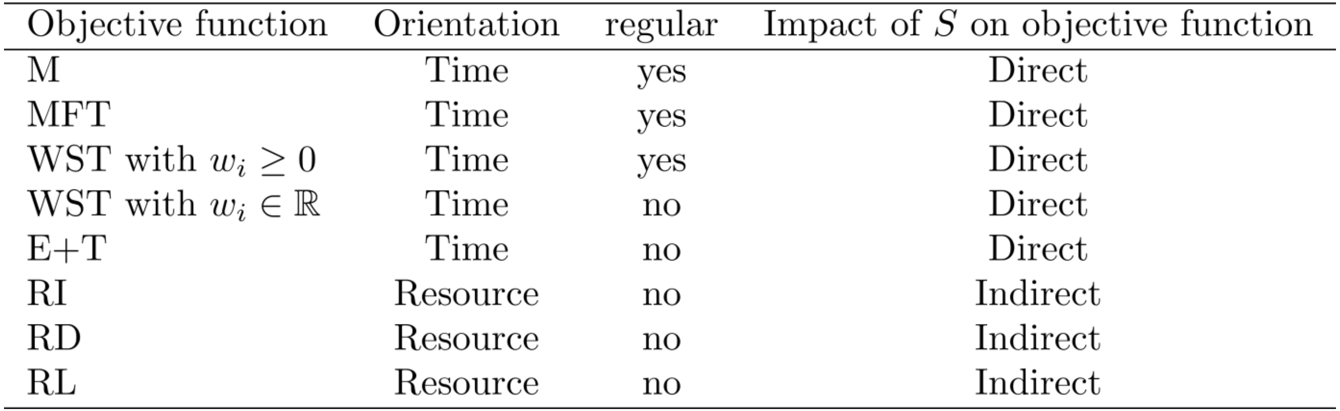
Resource Levelling Problem (**RL**): Min total weighted variation of resource demand

$$\text{Minimize } f(S) = \sum_{k \in R} c_k^L \sum_{t=0}^{T^{\max}} (\, r_k(S,t) \,)^2$$

with c_k^L weight for resource $k \in R$.

Remark: Scheduling problems with objective function RD, RI or RL are NP-hard optimization problems, even without resource constraints.

Characteristics of Objective Functions



Objective function	Orientation	regular	Impact of S on objective function
M	Time	yes	Direct
MFT	Time	yes	Direct
WST with $w_i \geq 0$	Time	yes	Direct
WST with $w_i \in \mathbb{R}$	Time	no	Direct
E+T	Time	no	Direct
RI	Resource	no	Indirect
RD	Resource	no	Indirect
RL	Resource	no	Indirect

Model for Scheduling with Time and Resource Constraints

Given a network $N = (V, E, \delta_{ij})$, the scheduling problem with time and resource constraints is:

make span or starti time

Min	$f(S)$		(1)
subject to	$S_j - S_i \geq \delta_{ij}$	for all $(i,j) \in E$	(2)
	$S_0 = 0$		(3)
	$r_k(S,t) \leq R_k$	for all $k \in R$, $t \in [0, T^{max})$	(4)

for each type of resource, for each period in planning horizon
resource demand less than resource supply
Can not throw to the solver since the input is the S already the objective variable

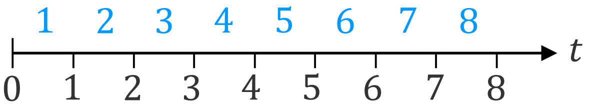
Remarks:

- Resource constraint (4) is due to $r_k(S,t)$ non linear and non convex.
- Model (1) – (4) is conceptual and cannot be implemented as LP or MIP.
- Problem (1) – (4) is a generalization of the Job Shop-Problems and thus NP-hard.

mathematical conceptual model
The problem

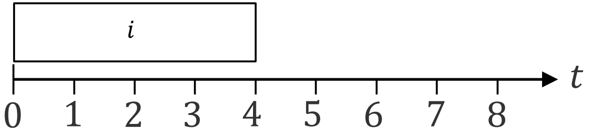
Time Points and Periods

- $[0, T^{max}]$ is the planning horizon
- We assume that minimum time lags $\delta_{i,j}$ and processing times p_i are (given as) integer (number of periods) and that $S_0 = 0$.
- Then, all activity start and finish time are integer.
- Period t begins at time $t-1$ and ends at time t
- We check capacity supply and demand of resources for period t at the start of the period, i.e. at time $t - 1$. That is, time period t is defined by the interval $[t-1, t)$.



periods

time points

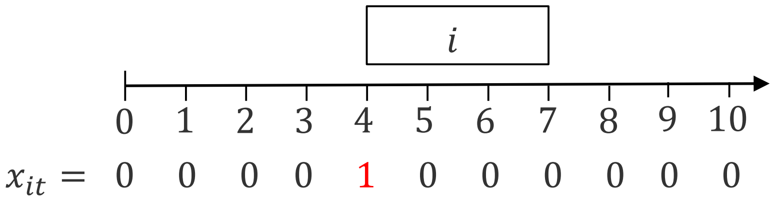


time points

Activity i with start time $S_i = 0$ and finish time 4. The activity is processed in periods $1, \dots, 4$.

Pulse Variables

$x_{it} =$ 1, if activity i is started at time t, 0 otherwise.



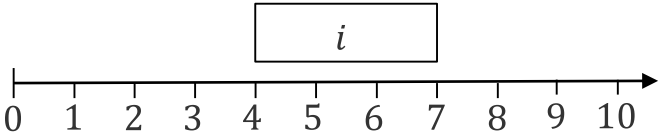
	0	1	2	3	4	5	6	7	8	9	10
$x_{it} =$	0	0	0	0	1	0	0	0	0	0	0

With the pulse variable we can derive the following:

- Start time of activity i: $\sum_{t=0}^{T^{\max}-1} t \cdot x_{i,t}$
- Finish time of activity i: $\sum_{t=0}^{T^{\max}-1} (t + p_i) \cdot x_{i,t}$
- Indicator function stating if activity i is processed at time t: $a_{i,t} = \sum_{\tau=t-p_i+1}^{t} x_{i,\tau}$ with new variable $a_{i,t} \geq 0$
- Demand of activity i for resource k at time t: $r_{i,k} \cdot a_{i,t} = r_{i,k} \sum_{\tau=t-p_i+1}^{t} x_{i,\tau} = \sum_{\tau=t-p_i+1}^{t} (r_{i,k} \cdot x_{i,\tau})$

Pulse Variables

$x_{it} =$ 1, if activity i is started at time t, 0 otherwise.

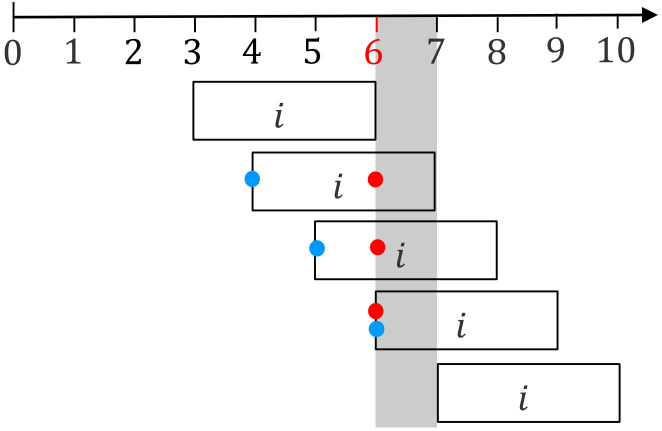


$x_{it} =$ 0 0 0 0 1 0 0 0 0 0 0

With the pulse variable we can derive the following:

- Start time of activity i: $\sum_{t=0}^{T^{\max}-1} t \cdot x_{i,t}$
- Finish time of activity i: $\sum_{t=0}^{T^{\max}-1} (t + p_i) \cdot x_{i,t}$
- Indicator function stating if activity i is processed at time t: $a_{i,t} = \sum_{\tau=t-p_i+1}^{t} x_{i,\tau}$ with new variable $a_{i,t} \geq 0$
- Demand of activity i for resource k at time t: $r_{i,k} \cdot a_{i,t} = r_{i,k} \sum_{\tau=t-p_i+1}^{t} x_{i,\tau} = \sum_{\tau=t-p_i+1}^{t} (r_{i,k} \cdot x_{i,\tau})$

Indicator Function stating if activity i is processed in period t



Consider time $t = 6$ and thus interval $[6,7)$.

Start times of activity i which cause the activity being active (processed) in $t = 6$ are 4, 5 and 6.

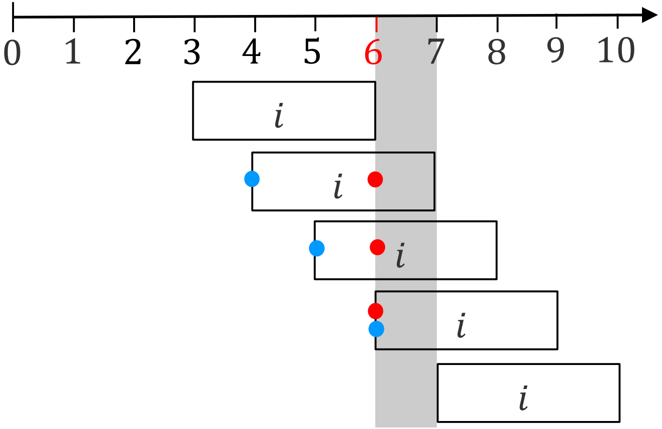
define the set of startime that leading to activities to time 6

In general, possible start times for being active in t are $S^a_{i,t} = f(t, p_i) = \{t - p_i + 1, \ldots, t\}$. with S

For $t = 6 : \; S^a_{i,6} = f(6,3) = \{6 - 3 + 1, \ldots, 6\} = \{4, \ldots, 6\}$

Indicator function stating if activity i is processed at time t: $a_{i,t} = \sum_{\tau \in S^a_{i,t}} x_{i,\tau} = \sum_{\tau = t - p_i + 1}^{t} x_{i,\tau}$

Demand of activity i for resource k at time t



Indicator function stating if activity i is processed at time t:

$$a_{i,t} = \sum_{\tau \in S^a_{i,t}} x_{i,\tau} = \sum_{\tau = t-p_i+1}^{t} x_{i,\tau}$$

Demand of activity i for resource k at time t:

$$r_{i,k} \cdot a_{i,t} = r_{i,k} \sum_{\tau = t-p_i+1}^{t} x_{i,\tau} = \sum_{\tau = t-p_i+1}^{t} (r_{i,k} \cdot x_{i,\tau})$$

Total Resource Demand at Time t with the Indicator Function

For resource $k \in R$ the total resource demand at time t is:

$$r_k(S,t) = \sum_{i \in A(S,t)} r_{i,k}$$

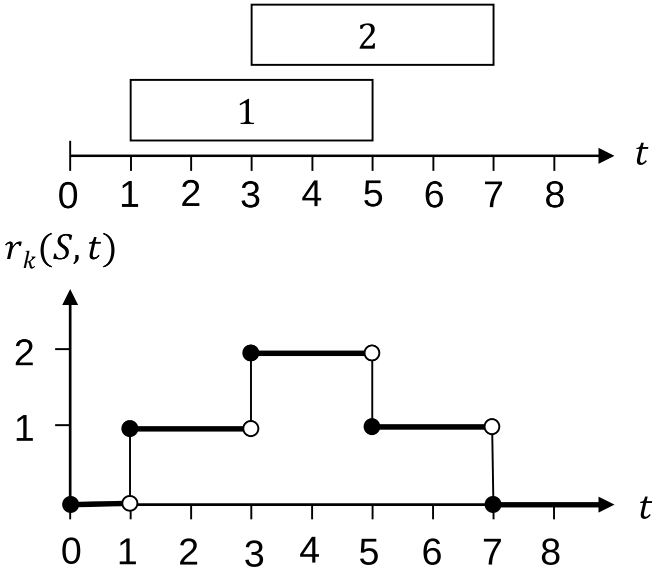
With the indicator function $a_{i,t} = f(x_{i,t})$ we can write:

$$\begin{aligned} r_k(\overset{x}{S},t) &= \sum_{i \in V} (r_{i,k} \cdot a_{i,t}) \\ &= \sum_{i \in V} \left(r_{i,k} \sum_{\tau = t - p_i + 1}^{t} x_{i,\tau} \right) = \sum_{i \in V} \sum_{\tau = t - p_i + 1}^{t} (r_{i,k} \cdot x_{i,\tau}) \end{aligned}$$

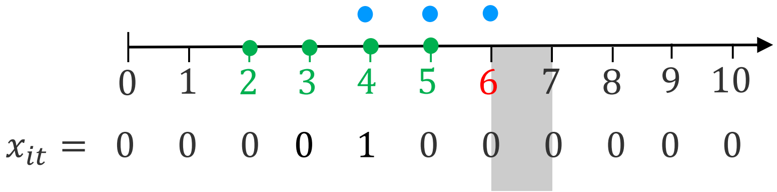
Example:

$S = (1,3),\ p = (4,4)$

$R = \{1\}, r_{i,1} = (1,1)$



Reducing the number of variables with time windows



$x_{it} = 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$

$ES_i = 2, LS_i = 5, W_i = \{2, \ldots, 5\}$

$S^a_{i,t=6} = \{4, \ldots, 6\}$

set of start time. due to blue dort and also bind the condition of start time

By forward and backward calculation we can derive time window $W_i = \{ES_i, \ldots, LS_i\}$ for the start times of activity i. With this we obtain:

- Start time of activity i: $\sum_{t \in W_i} t \cdot x_{i,t}$
- Finish time of activity i: $\sum_{t \in W_i} (t + p_i) \cdot x_{i,t}$

Taking into account the start time window, the set of start times of activity i for being active in t is $W_i \cap S^a_{i,t}$.

Indicator function stating if activity i is processed at time t: $a_{i,t} = \sum_{\tau \in \{W_i \cap S^a_{i,t}\}} x_{i,\tau} = \sum_{\tau = \max(ES_i, t - p_i + 1)}^{\min(t, LS_i)} x_{i,\tau}$

intersection between the time window and the set of active start time

RCPSP: Time-Indexed Formulation with Pulse-Variables

$$\text{Min } f(x) \qquad (1)$$

s.t.

$$\sum_{t \in W_i} x_{i,t} = 1 \qquad i \in V \qquad (2)$$

for each activity, start only once

$$\overset{\text{Sj}}{\sum_{t \in W_j} t \cdot x_{j,t}} - \overset{\text{Si}}{\sum_{t \in W_i} t \cdot x_{i,t}} \geq \delta_{ij} \qquad (i,j) \in E \qquad (3)$$

precedence contraint, for each arces has minimum time lag
Sj - Si >=Sij
now replace in each

$$\sum_{i \in V} \sum_{\tau = \max\{ES_i, t - p_i + 1\}}^{\min\{LS_i, t\}} r_{i,k} \cdot x_{i,\tau} \leq R_k \qquad k \in \mathcal{R},\ t \in \{0, \ldots, T^{\max} - 1\} \qquad (4)$$

Resouce constraints

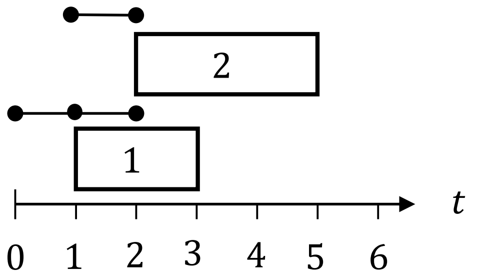
$$x_{i,t} \in \{0,1\} \qquad i \in V,\ t \in W_i \qquad (5)$$

- Min makespan objective function $f(x) = \sum_{t \in W_{n+1}} t \cdot x_{n+1,t}$
- Good LP bounds
- Number of binary variables is pseudo polynomial because the upper bound for the project duration equals the sum of the activity durations and the time window size of each activity grows with the upper bound of the project makespan. ⇒ Need for good upper bounds derived by heuristics.

Example Resource Constraints

Example: $|\mathcal{R}| = 1;\ T^{max} = 6$ $\qquad W_1 = \{0,1,2\};\ W_2 = \{1,2\}$

$r_{11} = 1;\ \ r_{21} = 1$ $\qquad p_1 = 2;\ p_2 = 3$



	$x_{1,0}$	$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$		
$t = 0$	1					$\leq$	R_1
$t = 1$	1	1		1		$\leq$	R_1
$t = 2$		1	1	1	1	$\leq$	R_1
$t = 3$			1	1	1	$\leq$	R_1
$t = 4$					1	$\leq$	R_1
$t = 5$						$\leq$	R_1

for example
Calculate the upper bounce of the Tmaz, just plus everything
T upper = sum pi = 9
we process the activities itself gradually, place each act after one another
it is already a feasible solution
Special simple precedent constraint

Now I want to find the solution with minimum makespan

first calculate time window: define the posible start time for tyhe activbities

i	0	1	2	3	4	
Esi	0	0	2	0	5	
Lsi	4	4	6	5	9	<- calculating backward
Wi	{0,4}	{0,4}	{2,6}	(0,5}	{5,9}	
|Wi|	5	5	5	6	5	-> 26 binanry variable

if we have longer horizon, then we have more binary variables
(possible to translate the problem to this one, by deviding to cont)

Disaggregated Precedence Constraints

Reference: Christofides, N., Alvarez-Valdes, R., and Tamarit, J.M. (1997): Project scheduling with resource constraints: A branch and bound approach, European Journal of Operational Research, Vol. 29 (3) 262-273.

Replace (3) by (3´)

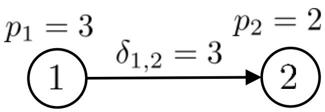
$$\sum_{t\in W_j} t\cdot x_{j,t} - \sum_{t\in W_i} t\cdot x_{i,t} \geq \delta_{i,j} \qquad \text{for all } (i,j)\in E \qquad (3)$$

$$\sum_{\tau=ES_i}^{\min(LS_i,t-\delta_{ij})} x_{i,\tau} - \sum_{\tau=ES_i}^{\min(LS_j,t)} x_{j,\tau} \geq 0 \qquad \text{for all } (i,j)\in E;\ t=0,\ldots,T^{\max}-1 \qquad (3')$$

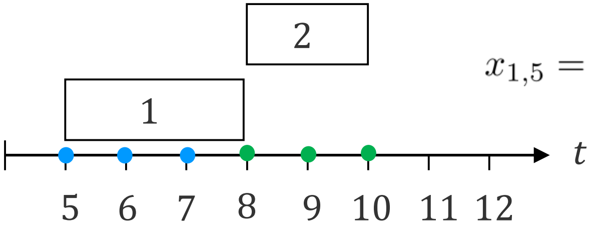
Constraint (3´) models the logical relations $S_j \leq t \Rightarrow S_i \leq t - \delta_{ij}$.

- Without resource constraints (4) the program has the unimodular property which means that the solution of the LP-relaxation gives an optimal solution for the MIP.
- Using (3´) gives better bounds for the LP-relaxation.
- (3´) has more constraints than (3): $|E| \cdot T^{max}$ instead of $|E|$.

Example: Disaggregated Precedence Constraints



$W_1 = \{5, 6, 7\}$ $W_2 = \{8, 9, 10\}$



$x_{1,5} = 1,\ x_{2,8} = 1$

$$\sum_{\tau=ES_i}^{\min(LS_i, t-\delta_{ij})} x_{i,\tau} - \sum_{\tau=ES_i}^{\min(LS_j, t)} x_{j,\tau} \geq 0 \qquad \text{for all } (i,j) \in E;\ t = 0, \ldots, T^{\max} - 1 \qquad (3')$$

For $t = 0, \ldots, 7$: $\sum_{\tau=5}^{4} x_{i,\tau} - \sum_{\tau=8}^{7} x_{j,\tau} \geq 0 \quad \Rightarrow$ No constraint

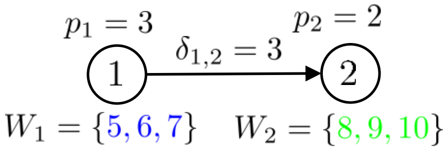
For $t = 8$: $\sum_{\tau=5}^{5} x_{i,\tau} - \sum_{\tau=8}^{8} x_{j,\tau} \geq 0 \quad \Rightarrow (x_{1,5}) - (x_{2,8}) \geq 0 \quad \Rightarrow 0 \geq 0$

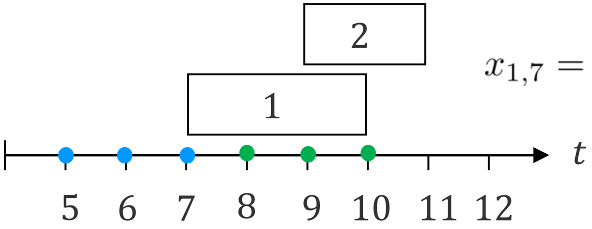
For $t = 9$: $\sum_{\tau=5}^{6} x_{i,\tau} - \sum_{\tau=8}^{9} x_{j,\tau} \geq 0 \quad \Rightarrow (x_{1,5} + x_{1,6}) - (x_{2,8} + x_{2,9}) \geq 0 \quad \Rightarrow 0 \geq 0$

For $t = 10$: $\sum_{\tau=5}^{7} x_{i,\tau} - \sum_{\tau=8}^{10} x_{j,\tau} \geq 0 \quad \Rightarrow (x_{1,5} + x_{1,6} + x_{1,7}) - (x_{2,8} + x_{2,9} + x_{2,10}) \geq 0$
$\Rightarrow 0 \geq 0$

For $t = 11, \ldots, T^{max} - 1$: $\sum_{\tau=5}^{7} x_{i,\tau} - \sum_{\tau=8}^{10} x_{j,\tau} \geq 0 \quad \Rightarrow (x_{1,5} + x_{1,6} + x_{1,7}) - (x_{2,8} + x_{2,9} + x_{2,10}) \geq 0$
$\Rightarrow 0 \geq 0$

Example: Disaggregated Precedence Constraints





$x_{1,7} = 1,\ x_{2,9} = 1$

$$\sum_{\tau=ES_i}^{\min(LS_i,t-\delta_{ij})} x_{i,\tau} - \sum_{\tau=ES_i}^{\min(LS_j,t)} x_{j,\tau} \geq 0 \qquad \text{for all } (i,j) \in E;\ t = 0, \ldots, T^{\max} - 1 \qquad (3')$$

For $t = 0, \ldots, 7$: $\sum_{\tau=5}^{4} x_{i,\tau} - \sum_{\tau=8}^{7} x_{j,\tau} \geq 0 \quad \Rightarrow$ No constraint

For $t = 8$: $\sum_{\tau=5}^{5} x_{i,\tau} - \sum_{\tau=8}^{8} x_{j,\tau} \geq 0 \quad \Rightarrow (x_{1,5}) - (x_{2,8}) \geq 0 \quad \Rightarrow 0 \geq 0$

For $t = 9$: $\sum_{\tau=5}^{6} x_{i,\tau} - \sum_{\tau=8}^{9} x_{j,\tau} \geq 0 \quad \Rightarrow (x_{1,5} + x_{1,6}) - (x_{2,8} + x_{2,9}) \geq 0 \quad \Rightarrow 0 - 1 \ngeq 0$

For $t = 10$: $\sum_{\tau=5}^{7} x_{i,\tau} - \sum_{\tau=8}^{10} x_{j,\tau} \geq 0 \quad \Rightarrow (x_{1,5} + x_{1,6} + x_{1,7}) - (x_{2,8} + x_{2,9} + x_{2,10}) \geq 0$
$\Rightarrow 1 - 1 \geq 0$

For $t = 11, \ldots, T^{max} - 1$: $\sum_{\tau=5}^{8} x_{i,\tau} - \sum_{\tau=8}^{10} x_{j,\tau} \geq 0 \quad \Rightarrow (x_{1,5} + x_{1,6} + x_{1,7} + x_{1,8}) - (x_{2,8} + x_{2,9} + x_{2,10}) \geq 0$
$\Rightarrow 1 - 1 \geq 0$

Lecture 6: MIP-Models for the RCPSP

Literature:

Artigues, C. (2013): A note on time-indexed formulations for the resource-constrained project scheduling problem, HAL-00833321, available at: https://hal.archives-ouvertes.fr/hal-00833321. (first publication on this topic)

Artigues. C., Koné, O., Lopez, P., Mongeau, M. (2015): Mixed-Integer Linear Programming Formulations, in Schwindt and Zimmermann (Eds.): Handbook on Project Management and Scheduling Vol. 1, Springer, Heidelberg, pp. 17 – 41. (follow up and extension of Artigues (2013), if you read only one of the three paper you should read this)

Artigues, C. (2017): On the strength of time-indexed formulations for the resource-constrained project scheduling problem, Operations Research Letters Vol. 45, pp. 154–159. (extends and corrects Artigues et al. (2015))

Survey of MIP-Formulations for the RCPSP

- Time-discrete models
  - Pulse-variables
  - Step variables
  - On-Off variables

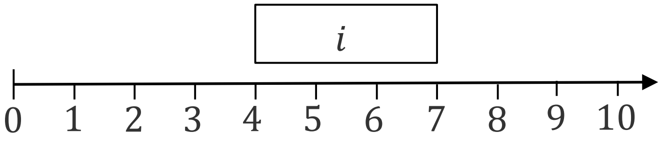
  Note:
  - Each model can be modelled with aggregate or disaggregate precedence constraints.
  - Each model can be transformed into each other model.

- Sequencing and continuous-time models
  - Flow-formulation

- Event-based models (not treated here)

Step Variable Formulation

$y_{it} = 1$, if activity i is started at time t or before, 0 otherwise.

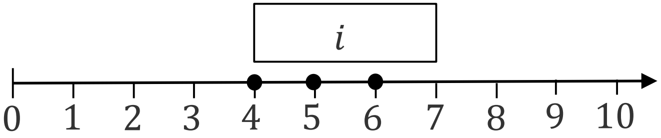


$y_{it} =$ 0 0 0 0 1 1 1 1 1 1 1

With the step variable we can derive the following:

- Start time of activity i: $\sum_{t=0}^{T^{\max}-1} t \cdot (y_{i,t} - y_{i,t-1})$
- Finish time of activity i: $\sum_{t=0}^{T^{\max}-1} (t + p_i) \cdot (y_{i,t} - y_{i,t-1})$
- Indicator function $a_{i,t}$ stating if activity i is processed at time t: $a_{i,t} = (y_{i,t} - y_{i,t-p_i})$
- Demand of activity i for resource k at time t: $r_{i,k} \cdot a_{i,t} = r_{i,k} \cdot (y_{i,t} - y_{i,t-p_i})$

Illustrations of Indicator Function for Step Variable



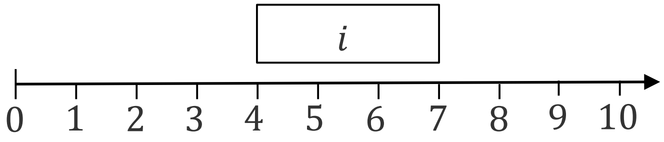
$y_{it} =$ 0 0 0 0 1 1 1 1 1 1 1

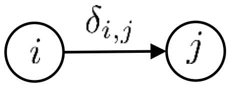
● = times when activity i is active

Indicator function $a_{i,t} = (y_{i,t} - y_{i,t-p_i})$

	y_{i0}	y_{i1}	y_{i2}	y_{i3}	y_{i4}	y_{i5}	y_{i6}	y_{i7}	
$(y_i) =$	0	0	0	0	1	1	1	1	
$t = 3$: $a_{i,3} =$	-1			1					$= 0$
$t = 4$: $a_{i,4} =$		-1			1				$= 1$
$t = 5$: $a_{i,5} =$			-1			1			$= 1$
$t = 6$: $a_{i,6} =$				-1			1		$= 1$
$t = 7$: $a_{i,7} =$					-1			1	$= 0$

Precedence Constraints for Step Variable



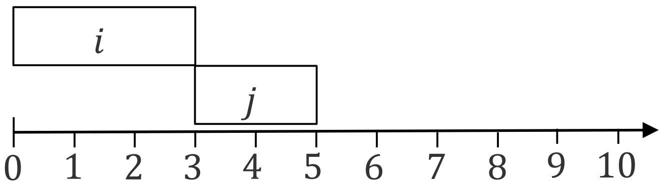


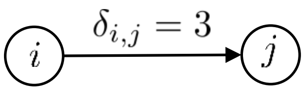
$y_{it} =$ 0 0 0 0 1 1 1 1 1 1 1

Disaggregated precedence constraint: $y_{i,t-\delta_{ij}} - y_{j,t} \geq 0$ for all $(i,j) \in E$ and for all $t = 0, \dots., T^{max}$

Aggregated precedence constraint: $\sum_{t=0}^{T} t \cdot (y_{j,t} - y_{j,t-1}) - \sum_{t=0}^{T} t \cdot (y_{i,t} - y_{i,t-1}) \geq \delta_{ij}$ for all $(i,j) \in E$

Disaggregated Precedence Constraints for Step Variables: Example





$$y_{it} = \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$$

Disaggregated precedence constraint: $y_{i,t-\delta_{ij}} - y_{j,t} \geq 0$ for all $(i,j) \in E$ and for all $t = 0, \ldots.., T^{max}$

$t = 0$: $y_{i,-3} - y_{j,0} \geq 0 \Rightarrow -y_{j,0} \geq 0 \Rightarrow y_{j,0} = 0$

$t = 1$: $y_{i,-2} - y_{j,1} \geq 0 \Rightarrow -y_{j,1} \geq 0 \Rightarrow y_{j,1} = 0$

$t = 2$: $y_{i,-1} - y_{j,2} \geq 0 \Rightarrow -y_{j,2} \geq 0 \Rightarrow y_{j2} = 0$

$t = 3$: $y_{i,0} - y_{j,3} \geq 0$

$t = 4$: $y_{i,1} - y_{j,4} \geq 0$

- Note:
  Since the t-indices are defined as $0 \leq t < T^{max}$, variables with negative t do not exist.
- Preprocessing would make the blue decisions.
- ~~Processing times are not required in the constraint~~

Step Variable Formulation with Disaggregated Precedence Constraints

Min $f(y)$ (1)

s.t.

$y_{i,t} = 0$ for all $i \in V$ and $t = 0, \ldots, ES_i - 1$ (2)

$y_{i,t} = 1$ for all $i \in V$ and $t = LS_i, \ldots, T^{max}$ (3)

$y_{i,t} - y_{i,t-1} \geq 0$ for all $i \in V$ and $t \in W_i$ (4)

$y_{i,t-\delta_{ij}} - y_{j,t} \geq 0$ for all $(i,j) \in E$ and for all $t = 0, \ldots.., T^{max}$ (5)

$\sum_{i \in V} r_{i,k} \cdot (y_{i,t} - y_{i,t-p_i}) \leq R_k$ for all $k \in \mathcal{R}$ and $t = 0, \ldots, T^{max} - 1$ (6)

$y_{it} \in \{0,1\}$ for all $i \in V$and $t \in W_i$ (7)

- Makespan objective: $f(y) = \sum_{t \in W_{n+1}} t \cdot (y_{n+1,t} - y_{n+1,t-1})$
- Activity time windows are taken into account in Constraints (2) and (3) prefixing all y_i-variables, which are not in the start time window $[ES_i, LS_i]$.
- ~~The step variable formulation provides tighter LP-bounds than the pulse variable formulation.~~

Step Variable Formulation with Aggregated Precedence Constraints

Min $\quad f(y)$ (1)

s.t.

$y_{i,t} = 0$ for all $i \in V$ and $t = 0, \dots, ES_i - 1$ (2)

$y_{i,t} = 1$ for all $i \in V$ and $t = LS_i, \dots, T^{max}$ (3)

$y_{i,t} - y_{i,t-1} \geq 0$ for all $i \in V$ and $t \in W_i$ (4)

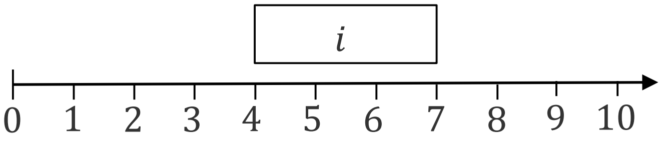
$\sum_{t=0}^{T} t \cdot \left(y_{j,t} - y_{j,t-1}\right) - \sum_{t=0}^{T} t \cdot \left(y_{i,t} - y_{i,t-1}\right) \geq \delta_{ij}$ for all $(i,j) \in E$ (5)

$\sum_{i \in V} r_{i,k} \cdot (y_{i,t} - y_{i,t-p_i}) \leq R_k$ for all $k \in \mathcal{R}$ and $t = 0, \dots, T^{max} - 1$ (6)

$y_{it} \in \{0,1\}$ for all $i \in V$and $t \in W_i$ (7)

On-Off Variable Formulation

$z_{it} = 1$, if activity i is in progress at time t, 0 otherwise.



$$z_{it} = \; 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0$$

On-Off Variable Formulation

With the on-off variable we can derive the following:

Let $\mathcal{T}_0 = \{0, \ldots, T^{\max} - 1\}$

- Start time of activity i with $p_i \geq 1$: $\sum_{t \in \mathcal{T}} t \cdot \left(\sum_{\alpha=0}^{\lfloor t/p_i \rfloor} z_{i,t-\alpha \cdot p_i} - \sum_{\alpha=0}^{\lfloor (t-1)/p_i \rfloor} z_{i,t-\alpha \cdot p_i - 1} \right)$
- Finish time of activity i: $\sum_{t \in \mathcal{T}_0} ((t + p_i) \cdot (y_{i,t} - y_{i,t-1}))$
- Indicator function $a_{i,t}$ stating if activity i is processed at time t: $z_{i,t}$
- Resource demand of activity i at time t: $r_{i,k} \cdot z_{i,t}$

On-Off-Based Formulation: Preprocessing

$$\phi(i) = \begin{cases} 1, & \text{if } p_i \geq 1 \\ 0, & \text{if } p_i = 0 \end{cases}$$

$$U(i) = \begin{cases} \{ES_i, \ldots, LF_i - 1\}, & \text{if } p_i \geq 1 \\ \{ES_i, \ldots, LS_i\}, & \text{if } p_i = 0 \end{cases}$$

$$T(i) = \begin{cases} \mathcal{T}_0 \setminus \{ES_i, \ldots, LF_i - 1\}, & \text{if } p_i \geq 1 \\ \{t \in \mathcal{T}_0 \mid t \leq ES_i - 1\}, & \text{if } p_i = 0 \end{cases}$$

$$K_{i,t} = \begin{cases} \lfloor t/p_i \rfloor, & \text{if } p_i \geq 1 \\ 0, & \text{if } p_i = 0 \end{cases}$$

On-Off-Based Formulation

$$\text{Min} \sum_{t \in \mathcal{T}} t \cdot (z_{n+1,t} - z_{n+1,t-1})$$

(1) Min makespan

s.t.

$$\sum_{\alpha=0}^{K_{i,t-p_i}} z_{i,t-(\alpha+1)\cdot p_i} - \sum_{\alpha=0}^{K_{j,t}} z_{j,t-\alpha\cdot p_j} \geq 0 \qquad (i,j) \in E,\ t \in \mathcal{T}_0$$

(2) Precedence constraints

$$\sum_{i \in V, p_i > 0} r_{i,k} \cdot z_{i,t} \leq R_k \qquad t \in \mathcal{T},\ k \in \mathcal{R}$$

(3) Resource constraints

$$\sum_{\alpha=0}^{K_{i,LF_i-\phi(i)}} z_{i,LF_i-\phi(i)-\alpha\cdot p_i} = 1 \qquad i \in V$$

(4) Each activity has to be performed

$$\sum_{\alpha=0}^{K_{i,t}} z_{i,t-\alpha\cdot p_i} - \sum_{\alpha=0}^{K_{i,t-1}} z_{i,t-\alpha\cdot p_i-1} \geq 0 \qquad i \in V,\ t \in \mathcal{T}$$

(5) No preemption

$$z_{i,t} = 0 \qquad i \in V,\ t \in T(i)$$

(6) Non-feasible processing periods

$$z_{i,t} \in \{0,1\} \qquad i \in V,\ t \in U(i)$$

(7) Feasible processing periods

Transformation of Variables

- The three time-discrete models can be transformed into each by variable transformation.

	$x_{i,t}$	$y_{i,t}$	$z_{i,t}$
$x_{i,t}$			$x_{i,t} = \sum_{\alpha=0}^{\lfloor t/p_i \rfloor} z_{i,t-\alpha \cdot p_i} - \sum_{\alpha=0}^{\lfloor (t-1)/p_i \rfloor} z_{i,t-\alpha \cdot p_i - 1}$
$y_{i,t}$			$y_{i,t} = \sum_{\alpha=0}^{\lfloor t \backslash p_i \rfloor} z_{i,t-\alpha \cdot p_i}$
$z_{i,t}$	$z_{i,t} = \sum_{\tau=t-p_i+1}^{t} x_{i,\tau}$	$z_{i,t} = y_{i,t} - y_{i,t-p_i}$	

Relationship between and Performance of Time-Discrete Models

- Each of the time discrete models can be formulated with disaggregated or aggregated precedence constraint.
- The LP-relaxations of the three formulations with disaggregated precedence constrains are equivalent.
- The LP-relaxations of the three formulations with aggregated precedence constraints are equivalent.
- For each model holds, the LP-relaxation of the variant with disaggregated precedence constraint is stronger than the aggregated precedence constraint.
- Empirical performance of the formulations is not necessarily related to the strength of the LP-relaxation. Good empirical results where obtained with the disaggregated step formulation.

Flow-Based Formulation

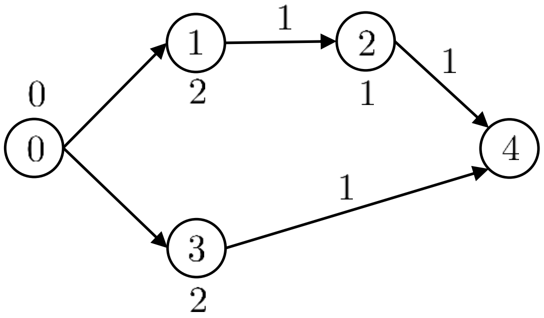
- First proposed by Artigues et al. (2003)

- Variables:
  - Start time variables: $S_i \geq 0$ for all $i \in V$.
  - Resource flow variables: $f_{ijk} \geq 0$; flow of resource units of type k from activity i to activity j.
  - Activity sequencing variables: $x_{ij} \in \{0,1\}$; 1, if $S_i + p_i \leq S_j$, 0 otherwise.

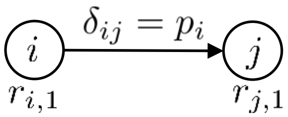
Note: The model employs only minimum time lags with $\delta_{ij} = p_i$.

Flow-Based Formulation: Example and Illustration

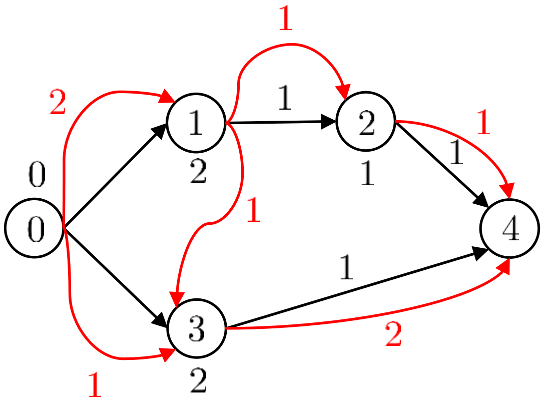
Problem:

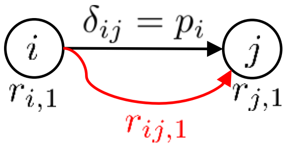


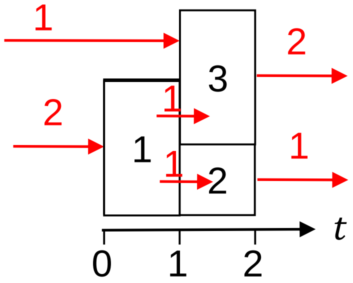
$| \mathcal{R} |= 1;\ R_1 = 3$



Solution:







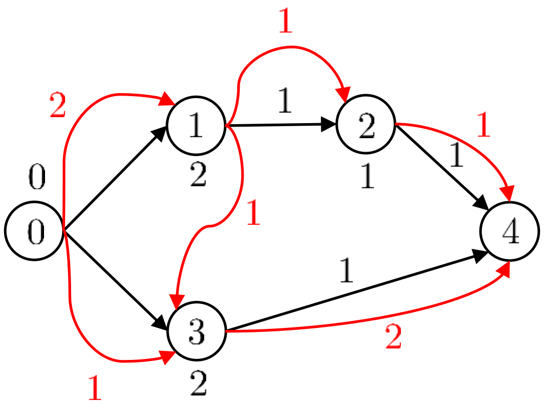
$S_0 = 0, S_1 = 0, S_2 = 1$
$S_3 = 1, S_4 = 2$

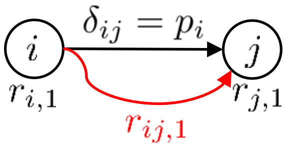
$x_{1,3} = 1$

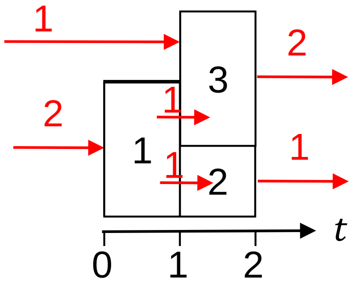
$r_{0,1,1} = 2, r_{0,3,1} = 1$
$r_{1,2,1} = 1, r_{2,4,1} = 1$
$r_{3,4,1} = 2,$

Transformation into Time-Constrained Scheduling Problem

Solution:







$$S_0 = 0, S_1 = 0, S_2 = 1$$
$$S_3 = 1, S_4 = 0$$

$$x_{1,3} = 1$$

$$r_{0,1,1} = 2, r_{0,3,1} = 1$$
$$r_{1,2,1} = 1, r_{2,4,1} = 1$$
$$r_{3,4,1} = 2,$$

After having obtained a feasible solution with the flow formulation, we can introduce for each $r_{ijk} > 0$ a precedence constraint $S_i + p_i \leq \ S_j$ in order to resolve the resource constraints. Hence, the new problem can be solved as time-constrained project scheduling problem by longest path calculations.

Flow-Based Formulation

$$\text{Min } S_{n+1} \qquad (1)$$

s.t.

$$x_{i,j} = 1 \qquad (i,j) \in E \qquad (2)$$

$$S_j - S_i - M \cdot x_{i,j} \geq p_i - M \qquad i \in V \backslash \{n+1\},\ j \in V \backslash \{0\} \qquad (3)$$

$$f_{i,j,k} - \min(r_{i,k}, r_{j,k}) \cdot x_{i,j} \leq 0 \qquad i \in V \backslash \{n+1\},\ j \in V \backslash \{0\},\ k \in \mathcal{R} \qquad (4)$$

$$\textstyle\sum_{j \in V \backslash \{0\}} f_{i,j,k} = r_{i,k} \qquad i \in V \backslash \{n+1\},\ k \in \mathcal{R} \qquad (5)$$

$$\textstyle\sum_{i \in V \backslash \{n+1\}} f_{i,j,k} = r_{j,k} \qquad j \in V \backslash \{0\},\ k \in \mathcal{R} \qquad (6)$$

$$x_{i,j} \in \{0,1\} \qquad i \in V \backslash \{n+1\},\ j \in V \backslash \{0\},\ i \neq j \qquad (7)$$

$$f_{i,j,k} \geq 0 \qquad i \in V \backslash \{n+1\},\ j \in V \backslash \{0\},\ i \neq j,\ k \in \mathcal{R} \qquad (8)$$

$$S_i \geq 0 \qquad i \in V \backslash \{0\} \qquad (9)$$

$$S_0 = 0 \qquad (10)$$

Flow-Based Formulation

(+): Polynomial number of variables and constraints.

(+): Suited for stochastic problem: x- and f-variables first stage, S-variables recourse.

(-): Poor LP relaxations.

Lecture 7: Modeling with Pulse-Variables

Literature: Rieck, J., J. Zimmermann, T. Gather (2012): Mixed-integer linear programming for resource leveling problems, European Journal of Operations Research, Vol. 221, pp. 27-37.

Modeling Time Objective Functions with Pulse Variables

Objective function M: $\sum_{t \in W_{n+1}} t \cdot x_{n+1,t}$

Objective function MFT: $\frac{1}{n+1} \sum_{i \in V} \sum_{t \in W_i} (t + p_i) \cdot x_{i,t}$

Objective function WST: $\sum_{i \in V} \sum_{t \in W_i} w_i \cdot t \cdot x_{i,t}$

Objective function E+T: $\sum_{i \in V} \left(c^E \cdot E_i + c^T \cdot T_i\right)$

$$\left.\begin{aligned} E_i &\geq d_i - \sum_{t \in w_i} (t + p_i) \cdot x_{i,t} \\ T_i &\geq \sum_{t \in w_i} (t + p_i) \cdot x_{i,t} - d_i \end{aligned}\right\} \text{for all } i \in V$$

$$E_i, T_i \geq 0$$

Modeling RCPSP-RI with Pulse Variables

$$\text{Min} \sum_{k \in \mathcal{R}} z_k \tag{1}$$

s.t.

$$\sum_{t \in W_i} x_{i,t} = 1 \qquad i \in V \tag{2}$$

$$x_{0,0} = 1 \tag{3}$$

$$\sum_{t \in W_j} t \cdot x_{j,t} - \sum_{t \in W_i} t \cdot x_{i,t} \geq \delta_{ij} \qquad (i,j) \in E \tag{4}$$

$$\sum_{i \in V} \sum_{\max\{ES_i, t-p_i+1\}}^{\min\{t, LS_i\}} r_{i,k} \cdot x_{i,\tau} \leq z_k \qquad k \in \mathcal{R},\ t \in \{0, \ldots, T^{\max} - 1\} \tag{5}$$

$$z_k \geq 0 \qquad k \in \mathcal{R} \tag{6}$$

$$x_{i,t} \in \{0,1\} \qquad i \in V,\ t \in W_i \tag{7}$$

Note: The objective function is non-regular. Hence we have to set the start time of activity 0 to 0 since this is not done through the objective function.

Modeling RCPSP-RD with Pulse Variables

Variable $z_{kt}^{+} \geq 0$ gives the demand for resource k at time t exceeding level Y_k

$$\text{Min} \sum_{k \in \mathcal{R}} \sum_{t \in \{0,\ldots,T^{\max}-1\}} c_k^D \cdot z_{k,t}^{+} \qquad (1)$$

s.t.

$$\sum_{t \in W_i} x_{i,t} = 1 \qquad i \in V \qquad (2)$$

$$x_{0,0} = 1 \qquad (3)$$

$$\sum_{t \in W_j} t \cdot x_{j,t} - \sum_{t \in W_i} t \cdot x_{i,t} \geq \delta_{ij} \qquad (i,j) \in E \qquad (4)$$

$$\sum_{i \in V} \sum_{\max\{ES_i, t-p_i+1\}}^{\min\{t, LS_i\}} r_{i,k} \cdot x_{i,\tau} - Y_k \leq z_{k,t}^{+} \qquad k \in \mathcal{R},\ t \in \{0,\ldots,T^{\max}-1\} \qquad (5)$$

$$z_{k,t}^{+} \geq 0 \qquad k \in \mathcal{R},\ t \in \{0,\ldots,T^{\max}-1\} \qquad (6)$$

$$x_{i,t} \in \{0,1\} \qquad i \in V,\ t \in W_i \qquad (7)$$

Note: The objective function is non-regular. Hence we have to set the start time of activity 0 to 0 since this is not done through the objective function.

Modeling RCPSP-RL with Pulse Variables

Upper Bound H_k of the demand for resource k: $H_k = \sum_{i \in V} r_{i,k}$

Variable $g_{kth} = 1$, if the demand for resource k at time t is h, 0 otherwise:

$$\text{Min} \sum_{k \in \mathcal{R}} c_k^L \sum_{t \in \{0,\ldots,T^{\max}-1\}} \sum_{h=0}^{H_k} h^2 \cdot g_{k,t,h} \tag{1}$$

s.t.

$$\sum_{t \in W_i} x_{i,t} = 1 \qquad i \in V \tag{2}$$

$$x_{0,0} = 1 \tag{3}$$

$$\sum_{t \in W_j} t \cdot x_{j,t} - \sum_{t \in W_i} t \cdot x_{i,t} \geq \delta_{ij} \qquad (i,j) \in E \tag{4}$$

$$\sum_{i \in V} \left(r_{i,k} \sum_{\max\{ES_i, t-p_i+1\}}^{\min\{t, LS_i\}} x_{i,\tau} \right) \leq \sum_{h=0}^{H_k} h \cdot g_{k,t,h} \qquad k \in \mathcal{R},\ t \in \{0,\ldots,T^{\max}-1\} \tag{5}$$

$$\sum_{h=0}^{H_k} g_{k,t,h} = 1 \qquad k \in \mathcal{R},\ t \in \{0,\ldots,T^{\max}-1\} \tag{6}$$

$$g_{k,t,h} \in \{0,1\} \qquad k \in \mathcal{R},\ t \in \{0,\ldots,T^{\max}-1\};\ h \in \{0,\ldots,H_k\} \tag{7}$$

$$x_{i,t} \in \{0,1\} \qquad i \in V,\ t \in W_i \tag{8}$$

Application: Maintenance of Nuclear Power Plants

Reference: Rieck, J, Zimmermann, J. and Gather, T. (2012): Mixed-integer linear programming for resource leveling problems, European Journal of Operational Research, Vol. 221, 27-37.

- Rieck et al. (2012) pp. 29 – 30.
- One maintenance per year.
- Approx. 20 – 50 work packages with 5,000 activities.
- Internal and approx. 1,000 external workers.
- Independent experts supervise the processing of the work packages.
- Cooling of the reactor and the fuel rod container has to be guaranteed. Only one of the cooling system can be shut down for maintenance at a time. No power generation during maintenance. Loss of revenue of approx. 1 million EURO per day.

Application: Maintenance of Nuclear Power Plants

- Areas where workers are working in parallel are narrow, in particular annular catwalks in the containment which are providing access to a large number of pipes and valves which need to be inspected. No more than 15-20 workers. In case of an accident, a small number of workers in confined areas allows safe and orderly evacuations
- Renewable resources: Workers, supervisors, confined areas.
- First Objective: Minimization of down time (≤ 4 weeks).
- Second objective (with first objective fixed as constraint): RL- or RD-objective function (with regular capacity of R=15 workers).

Lecture 8: Schedule Generation Schemes

References:

Kolisch, R. (1996): Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation, European Journal of Operational Research, Vol. 90, pp. 320-333.

Kolisch, R. und S. Hartmann (1999): Heuristic algorithms for solving the resource-constrained project scheduling problem - Classification and computational analysis, in Weglarz, J. (Eds.): Project Scheduling – Recent Models, Algorithms and Applications, Kluwer, Boston, pp. 147-178.

Kolisch, R. (2015): Project Schedules: Shifts, Types, and Generation Schemes, in: Schwindt, C. and J. Zimmermann (Eds.): Handbook on Project Management and Scheduling Vol. 1, Springer, Berlin, pp. 3–16.

Note:

- Schedule generation schemes (SGS) were originally devised for the RCPSP with minimum time lags $\delta_{ij} = p_i$ and objective function M. This is what we assume in the following.

Serial Schedule Generation Scheme

Notation:

$C \subseteq V$: Set of activities, which have been scheduled

$Pred(j) = \{i \in V \mid (i,j) \in A\}$: Set of activities, which are immediate predecessors of activity j

$D = \{i \in V \setminus C \mid Pred(i) \in C\}$: Decision set. Set of activities, which can be scheduled

$A(C,t) = \{i \in C \mid S_i \leq t < S_i + p_i\}$: Set of activities, which are active at time t

$\tilde{R}_k(t) = R_k - \sum_{i \in A(C,t)} r_{i,k}$: Remaining capacity of resource r at time t

Initialization: $S_0 = 0$, $C_0 = \{0\}$

Iteration:

```
For $g = 1$ to $n$:
    Update $D$, $\tilde{R}_k(t)$ for all $k \in R$ and $t = 0, \ldots, T-1$
    Select one $j \in D$
    $\overline{ES}_j = \max\{S_i + p_i \mid i \in Pred(j)\}$
    $S_j = \min\{t \mid \overline{ES}_j \leq t,\ r_{j,k} \leq \tilde{R}_k(\tau)$ for all $k \in R$ and $\tau = t, \ldots, t + p_j - 1\}$
    $C = C \cup \{j\}$
$S_{n+1} = \max\{S_i + p_i \mid i \in Pred(n+1)\}$
```

Serial Schedule Generation Scheme with iteration counter n

Notation:

n: Iteration counter

$C_n \subseteq V$: Set of activities, which have been scheduled at the end of iteration n

$Pred(j) = \{i \in V \mid (i,j) \in A\}$: Set of activities, which are immediate predecessors of activity j

$D_n = \{i \in V \setminus C \mid Pred(i) \in C\}$: Decision set. Set of activities, which can be scheduled in iteration n

$A(C_n, t) = \{i \in C \mid S_i \leq t < S_i + p_i\}$: Set of activities, which are active at time t

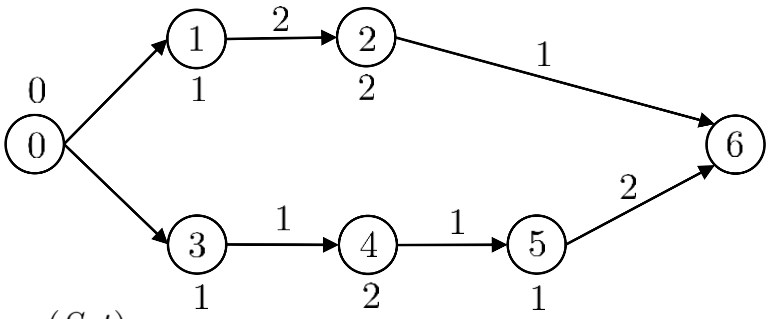
$\tilde{R}_k(t) = R_k - \sum_{i \in A(C_n,t)} r_{i,k}$: Remaining capacity of resource r at time t

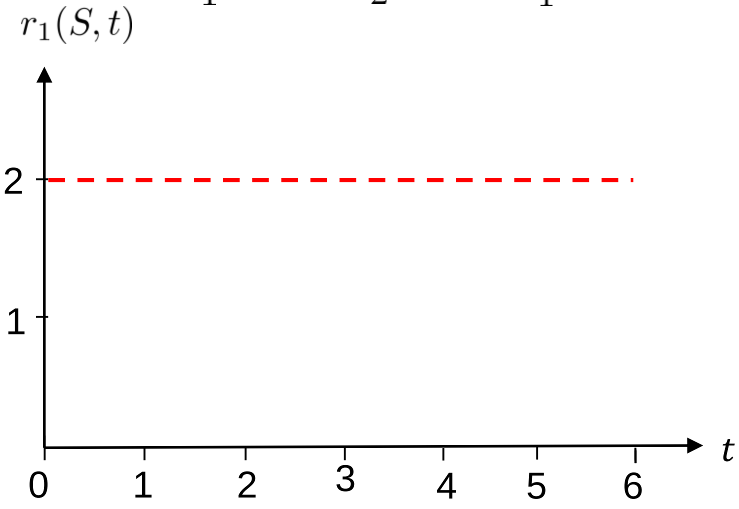
Initialization: $S_0 = 0$, $C_0 = \{0\}$

Iteration:

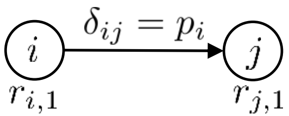
```
For $g = 1$ to $n$:
    Update $D_n$
    Select one $j \in D_n$
    $\overline{ES}_j = \max\{S_i + p_i \mid i \in Pred(j)\}$
    $S_j = \min\{t \mid \overline{ES}_j \leq t,\ r_{j,k} \leq \tilde{R}_k(\tau) \text{ for all } k \in R \text{ and } \tau = t, \ldots, t + p_j - 1\}$
    $C_n = C_{n-1} \cup \{j\}$
$S_{n+1} = \max\{S_i + p_i \mid i \in Pred(n+1)\}$
```

Serial Schedule Generation Scheme: Example





$|\mathcal{R}| = 1;\ R_1 = 2$



Priority rule: Min slack

j	0	1	2	3	4	5	6
Slack							

g	C	D	j	ES_j	t

Serial Schedule Generation Scheme: Characteristics

- Complexity is $O(n^2 \cdot \mid \mathcal{R} \mid)$
- If there exists a feasible solution for objective function M, the serial SGS will generate a feasible solution.
- Generates active schedules, i.e. no activity can be left shifted without performing a right shift of one or more other activities. For a regular objective function, an optimal schedule is always active.
- Generates an activity list $L \ = < j_0, j_1, \dots, j_{n+1}]$ where j_k is the activity selected and scheduled in iteration k. Activity list L has property $Pred(j_k) \subseteq \{j_0, j_2, \dots, j_{k-1}\}$, that is all predecessors of activity j_k are in front of j_k in the list. A list L can be mapped into a schedule S using the serial schedule generation scheme.
- For objective function M, there is always one list L which gives an optimal solution when mapped into a schedule with the serial SGS. Accordingly, there exist one “optimal” priority rule for the serial SGS, which can be obtained in retrospect.
- Problem with detecting optimal solutions when applying heuristics. One way: Use of lower bounds.

Parallel Schedule Generation Scheme

Notation:

t_g: Schedule time at iteration g

$C_g = \{i \in V \mid S_i + p_i \leq t_g\}$: Set of completed activities

$A_g = \{i \in V \mid S_i \leq t_g < S_i + p_i\}$: Set of active activities

$R_k(t_g) = R_k - \sum_{i \in A_g} r_{i,k}$: Available capacity at t_g

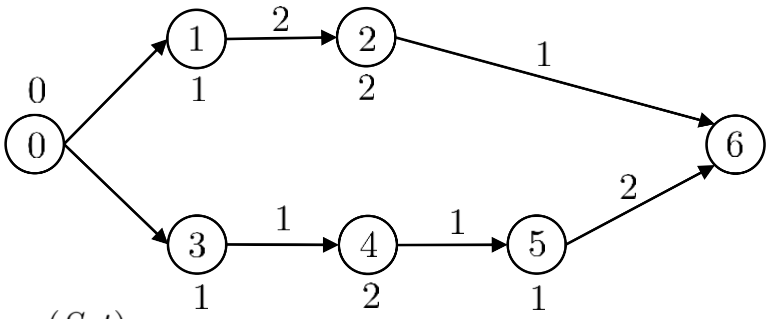
$D_g = \{i \in V \setminus \{C_g \cup A_g\} \mid Pred(i) \in C_g, r_{i,k} \leq R_k(t_g)$ for all $k \in R\}$: Decision set

Initialization: $g = 0,\ t_0 = 0,\ S_0 = 0,\ A_0 = \{0\},\ C_0 = \{\}$

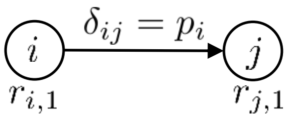
Iteration:

```
While $| A_g \cup C_g | \leq n$:
    $g = g + 1$
    $t_g = \min\{S_j + p_j \mid j \in A_{g-1}\}$
    Calculate $C_g$, $A_g$, $R_k(t_g)$ for all $k \in \mathcal{R}$, $D_g$
    While $D_g \neq \{\}$:
        Select one $j \in D_g$
        $S_j = t_g$
        Calculate $R_k(t_g)$ for all $k \in \mathcal{R}, A_g, D_g$
$S_{n+1} = \max\{S_i + p_i \mid i \in Pred(n+1)\}$
```

Parallel Schedule Generation Scheme: Example



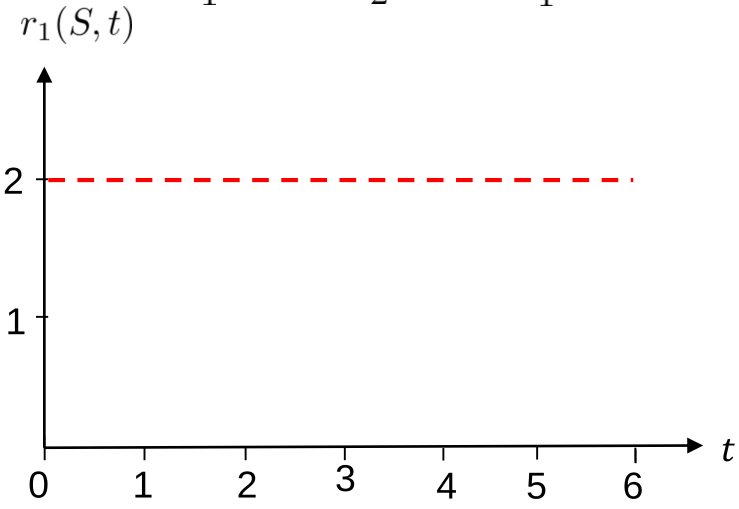
$| \mathcal{R} |= 1;\ R_1 = 2$



Priority rule: Min slack

j	0	1	2	3	4	5	6
Slack							

g	t_g	C_g	A_g	$R_1(t_g)$	D_g	j	S_j



Parallel Schedule Generation Scheme: Characteristics

- Complexity O($n^2|R|$)
- If there exists a feasible solution for objective function M, the parallel SGS will generate a feasible solution.
- The parallel SGS scheme generates nondelay schedules. A nondelay schedule is a schedule where when each activity j with duration p_j is split up into p_j activities with duration 1, no activity can be started earlier without at least one other activity starting later.
- The cardinality of the set of nondelay schedules is much smaller than the cardinality of the set of active schedules. Usually, for regular objective functions, the best nondelay schedules are very good. However, there might not be an optimal schedule within the set.
- Kolisch (1996) has empirically shown that 60% of the instances contain an optimal nondelay schedule for objective function M.
- The parallel SGS generates an activity list $L\ = < j_0, j_1, \dots, j_{n+1}]$ where j_k is the activity selected and scheduled in iteration k. Activity list L has property $Pred(j_k) \subseteq \{j_0, j_2, \dots, j_{k-1}\}$, that is all predecessors of activity j_k are in front of j_k in the list. A list L can be mapped into a schedule S using the parallel SGS.

Lecture 8: Lower Bounds for the RCPSP

- Purpose of lower bounds.

<u>Two simple lower bounds</u>:

- Precedence-based lower bound: $LB_1 = ES_{n+1}$

- Resource-based lower bound: $LB_2 = \max\limits_{r \in \mathcal{R}} \lceil \sum\limits_{i \in V} \frac{p_i \cdot r_{i,k}}{R_k} \rceil$

<u>More elaborated lower bounds</u>:

- LP-based constructive lower bound: Mingozzi et al. (1998)
- LP-based destructive lower bound: Brucker und Knust (2000)
- Lagrange-based lower bound: Möhring et al. (2003), Bianco and Caramia (2011)

Lecture 8: Constructive and Sampling Heuristics

Single-Pass Priority Rule Heuristics

- Single priority rule and single schedule generation scheme → Single solution
- Many priority rules have been proposed in the literature (e.g. SPT, LFT, LST, MSLK). Priority rules based on lower bounds such as LFT show the best results.
- Quality of solutions for good priority rules: For projects with 30 activities and 4 resources, good rules will be about 5% above the optimum. Conjecture: Gap to optimal solution increases with problem size.

Deterministic Multi-Pass Priority Rule Heuristics

- Combination of different priority rules, schedule generation schemes and forward and backward scheduling (reversed network) → Multiple solutions, some of them might be identical. Select the best solution.

Linear combination of priority values stemming from different priority rules

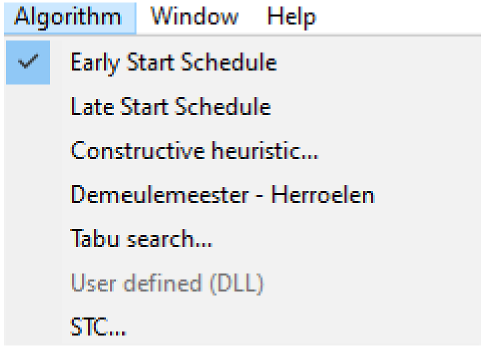
- Comparing apple and oranges.
- Normalization by, e.g., using ranks instead of priority values.

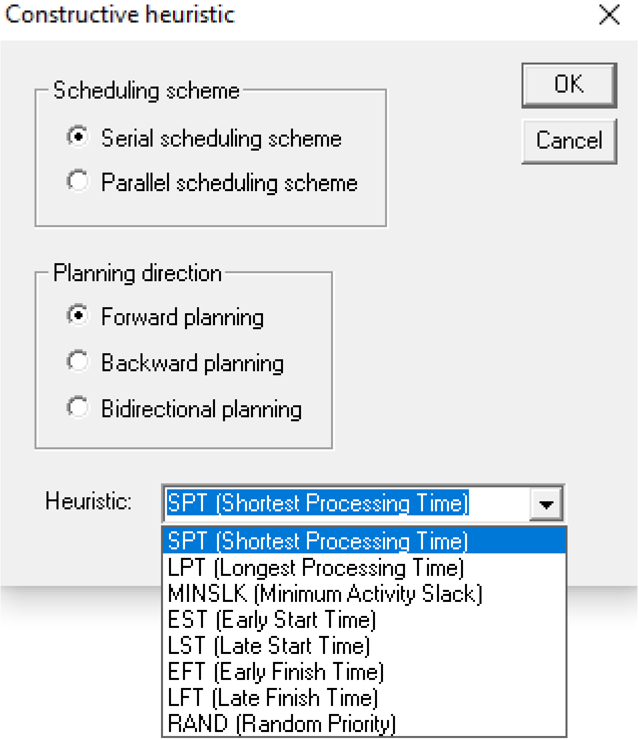
Software RESCON

Reference: Deblaere, F, Demeulemeester, E. und Herroelen, W. (2011): RESCON: Educational project scheduling software, Computer Applications in Engineering Education, Vol. 19 (2), 327-336.

- Available at http://www.econ.kuleuven.be/rescon/
- Generation of project schedules with customizable priority rule heuristics (SGS, priority rules, scheduling direction), an exact algorithm and a Tabu Search heuristic.
- Graphical representation of the makespan determined by all algorithms
- Graphical representation of project network, project schedule, and resource Gantt chart.

RESCON: Constructive Heuristics





Constructive heuristic

Scheduling scheme
- ● Serial scheduling scheme
- ○ Parallel scheduling scheme

Planning direction
- ● Forward planning
- ○ Backward planning
- ○ Bidirectional planning

Heuristic: SPT (Shortest Processing Time)

SPT (Shortest Processing Time)
LPT (Longest Processing Time)
MINSLK (Minimum Activity Slack)
EST (Early Start Time)
LST (Late Start Time)
EFT (Early Finish Time)
LFT (Late Finish Time)
RAND (Random Priority)

RESCON Data Format

$n + 2,\ K$

$R_1, R_2, \ldots, RK$

$p_0, 0{,}0, \ldots, 0, |S_0|$, direct successors separated by comma

$p_1, r_{1,1}, r_{1,2}, \ldots . ., r_{1,K}, |S_1|$, direct successors separated by comma

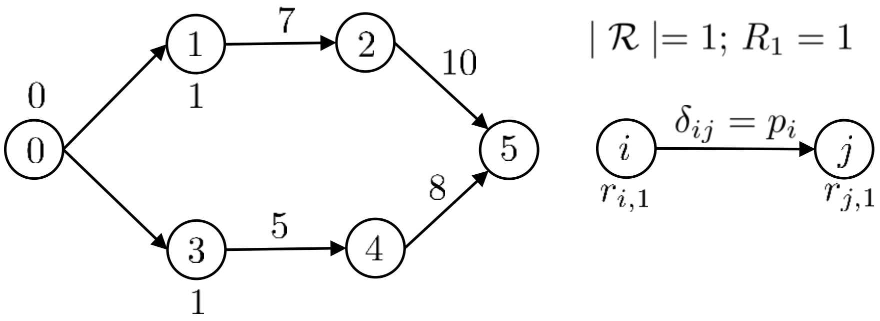
$\ldots$

$p_{n+1}, 0{,}0, \ldots, 0, |S_{n+1}|$

Note: RESCON numbers activities from 1 to $n + 1$.

Example:

$6, 1$
1
$0, 0, 2, 2, 4$
$7, 1, 1, 3$
$10, 0, 1, 6$
$5, 1, 1, 5$
$8, 0, 1, 6,$
$0, 0, 0$



Sampling Heuristics

- In each iteration of one of the SGS, activity j is selected probabilistically instead of deterministically.
- SGS can be applied several times, generating potentially different solutions. Selection of the best solution.
- $prob(j)$ = selection probability of activity $j \in D$
- Selection probabilities can be derived based on the priority values obtained with a priority rule.
- Most simple sampling rule: Random with $prob(j) = |D|^{-1}$
- Different ways of calculating probabilities. One approach is Regret-Based Biased Random Sampling (RBRS).

Regret-Based Biased Random Sampling

Reference: Kolisch, R. und A. Drexl (1996): Adaptive search for solving hard project scheduling problem, Naval Research Logistics, Vol. 43, S. 23-40.

Regret: $\rho_j = \max_{i \in D}\{LF_i\} - LF_j + 1$

Selection probability: $Prob(j) = \frac{\rho_j^\alpha}{\sum_{i \in D} \rho_i^\alpha}$

Example:

$D = \{1{,}2\}$

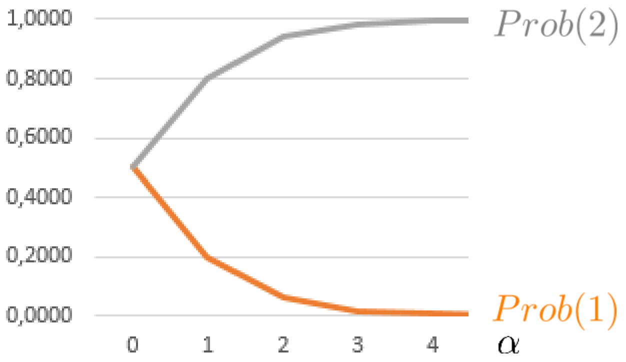
Priority values (LFT): $LF_1 = 7, LF_2 = 4$

$\rho_1 = 1, \rho_2 = 4$

$\alpha = 1$: $Prob(1) = 0.2$, $Prob(2) = 0.8$

$\alpha = 0$: $Prob(1) = 0.5$, $Prob(2) = 0.5$ (Random sampling)

$\alpha = M$: $Prob(1) = 0$, $Prob(2) = 1$ (Deterministic selection)



Lecture 9: Improvement Procedures and Metaheuristics

Improvement Procedures

- General idea
  - Given a feasible solution, we try to improve the solution by an algorithm until it cannot be improved by this algorithm any further.
  - The new solution obtained is called a local optimum.
- Here, we will treat the Forward-Backward-Improvement (FBI) procedure, which is also named Justification.

References:

FBI: P. Tormos and A. Lova (2001): A Competitive Heuristic Solution Technique for Resource-Constrained Project Scheduling, Annals of Operations Research, Vol. 102, pp. 65-81.

Justification: V. Valls, F. Ballestin and S. Quintanilla (2005): Justification and RCPSP: A technique that pays, European Journal of Operational Research, Vol. 165, pp. 375-386.

Forward-Backward Improvement Procedure

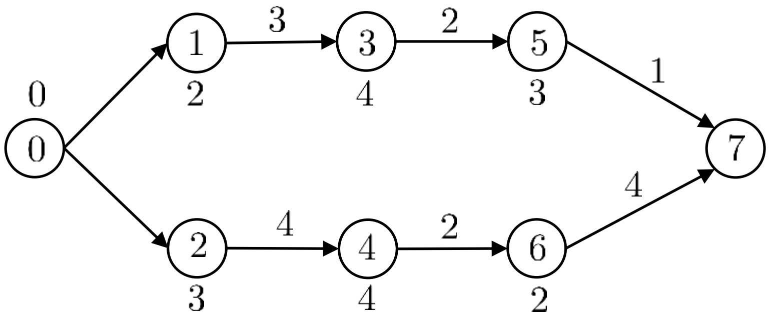
Start with a feasible schedule generated with the serial or parallel schedule generations scheme.
Note: Such a schedule is (left-active), i.e., no activity can be started earlier without delaying another activity.

Backward Scheduling: Derive an activity list of the reversed network by sorting activities according to descending finish times of the previous schedule. The first activity in the list is activity $n + 2$. Set the start time of activity $n + 2$ to its start time in the previous schedule. Using the serial SGS right-schedule the activities of the reversed network according to the list.
Result: Right-active schedule with makespan ≤ makespan of the former schedule.

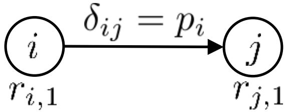
Forward Scheduling: Derive an activity list of the network by sorting the activities according to ascending start times of the previous schedule. The first activity in the list is activity 0. Set the start time of activity 0 to 0 and left-schedule the activities in the order of the activity list.
Result: Left-active schedule with a makespan ≤ makespan of the former schedule.

Stop when the makespan is not further decreased.

Forward-Backward Improvement: Example



$| \mathcal{R} | = 1;\ R_1 = 4$

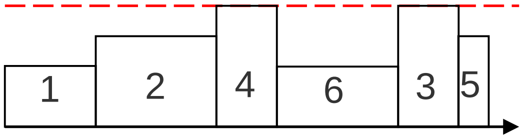


1st Forward:

<0,1,2,4,6,3,5,7]

+

Serial SGS



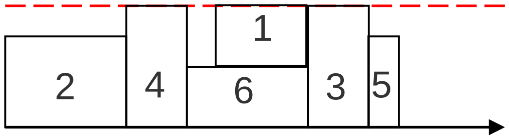
S=(0,0,3,13,7,15,9,16)

1st Backward:

[0,1,2,4,6,3,5,7>

+

Serial SGS



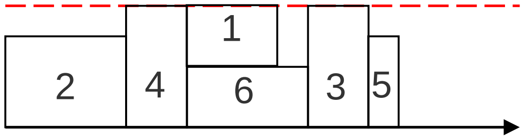
F=(0,10,4,12,6,13,10,13)

2nd Forward:

<0,2,4,6,1,3,5,7]

+

Serial SGS



S=(0,6,0,10,4,12,6,13)

Metaheuristics for the RCPSP

Metaheuristics are solution methods that orchestrate an interaction between local improvement (= local search) procedures and higher level strategies in order to escape from local optima (Glover and Kochenberger, 2003).

Local Search is an iterative procedure which moves from one solution in the set of feasible solutions to another until a stopping criterion is satisfied. In order to move systematically through the solution set, a neighborhood structure is used which defines for each solution a set of neighborhood solutions (Brucker and Knust, 2012).

Gendreau and Potvin (2019) provide a general presentation of the main metaheuristics.

References:

- Brucker, P. und Knust, S. (2012): Complex Scheduling, 2rd Ed., Springer.
- Gendreau M. and Potvin, J.-Y. (2019): Handbook of Metaheuristics, 3rd Ed., Springer, New York.
- Glover, F. and Kochenberger, G.A. (2003): Handbook of Metaheuristics, Springer, New York.

Metaheuristics for the RCPSP/M

A plethora of metaheuristics for the RCPSP (as for many other NP-hard optimization problems such as the VRP) has been proposed in the literature. A categorization and surveys are given in Kolisch and Hartmann (1999, 2006) and Hartmann and Kolisch (2000). Here we will discuss the following two metaheuristics:

- Genetic Algorithm (GA) of Hartmann (2002)
- Simulated Annealing (SA) of Bouleimen and Lecocq (2003)

References:

Kolisch, R. and Hartmann, S. (2006): Experimental investigation of heuristics for resource-constrained project scheduling: An update, European Journal of Operational Research, Vol. 174(1) 23-37.

Hartmann, S. (2002): A self-adapting genetic algorithm for project scheduling under resource constraints, Naval Research Logistics, Vol. 49, pp. 433-448.

Bouleimen, K. and Lecocq, H. (2003): A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, European Journal of Operational Research, Vol. 149, pp. 268-281.

Genetic Algorithm: General Framework

- Proposed by Holland in 1975. General description see, e.g., Chapter 5 by Reeves in in Gendreau and Potvin (2019).

- Mimics the evolution as proposed by Darwin.

- Number of generations (parameter).

- For each generation there is a set (pool) of solutions. The size of the population is a parameter which has to be determined appropriately.

- Each solution is coded by using some form of representation. The coded solution is called “genotype” and the full solution is called “phenotype”. The genotype is mapped with a function into the phenotype. For each coded solution we have the objective function value (fitness function value).

Genetic Algorithm: General Framework

- A given number of solution pairs (mother and father) are randomly selected from the current generation.
  Note: There are alternative selection methods.

- For each pair of solutions, two new solutions (“son” and “daughter”) are generated by crossover.

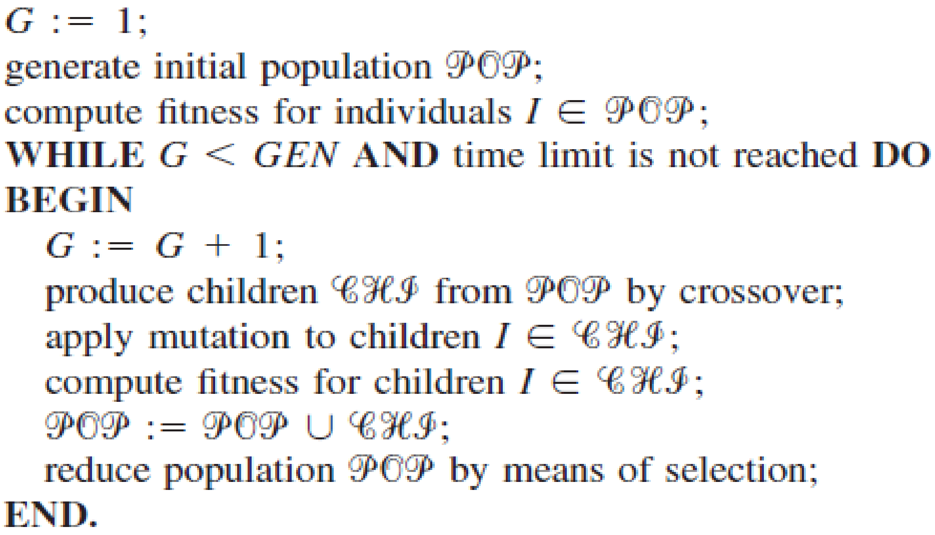
- With some probability, each new solution is undergoing mutation, a slight change of the genotype. The phenotype and objective functions for the son and the daughter are generated, respectively.

- Merging the parent-generation with the daughter-and-son generation (doubles the size of the population).

- Reduce the population to its original size by selecting individuals.
  Note: There are alternative selection methods available.

- Start with the next generation.

Genetic Algorithm: Pseudocode



```
$G := 1$;
generate initial population $\mathcal{POP}$;
compute fitness for individuals $I \in \mathcal{POP}$;
WHILE $G < GEN$ AND time limit is not reached DO
BEGIN
    $G := G + 1$;
    produce children $\mathcal{CHI}$ from $\mathcal{POP}$ by crossover;
    apply mutation to children $I \in \mathcal{CHI}$;
    compute fitness for children $I \in \mathcal{CHI}$;
    $\mathcal{POP} := \mathcal{POP} \cup \mathcal{CHI}$;
    reduce population $\mathcal{POP}$ by means of selection;
END.
```

Details of Hartmann’s Genetic Algorithm

Building blocks:

- Solution representation: Activity list
- Generations of initial population:
  - Regret-based biased random sampling with LFT- and LST-priority rule.
  - Probabilistic choice of schedule generation scheme with equal probabilities.
- Selection of parent solutions: Random
- Crossover: 2-point crossover
- Mutation
- Selection of next-generation (surviving) solutions: Select individuals according to increasing objective function value (makespan). “Surviving of the fittest”.

Further Details:

- Population size = 40, number of generations = 25 → 1,000 generated solutions

Solution Representation: Activity List

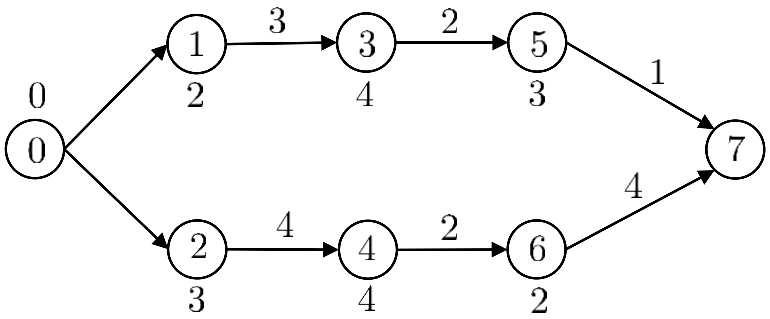
$L \ = < j_0, j_1, \dots, j_{n+1}]$ where j_j is activity at position $(j+1)$ of the list.

Sequence of the $n+2$ activities $0, \dots, n+1$. Activity 0 is always at position 1 and activity $n+1$ is always at position $n+2$, i.e., $j_0 = 0$ and $j_{n+1} \ = n+1$. The activity sequence has to respect the topological order of the network, i.e., $Pred(j_n) \subseteq \{j_1, j_2, \dots, j_{n-1}\}$ has to hold for all $0 \leq n \leq |V|$.

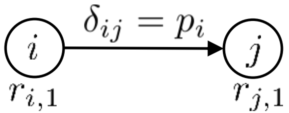
Remember: Using a priority rule and a schedule generation scheme yields a feasible activity list.

Each activity list can be mapped with the parallel or the serial schedule generation scheme into a schedule. Note that the mapping is not unique. I.e., different activity lists can lead to the same schedule.

Genetic Algorithm: Solution Representation



$|\mathcal{R}| = 1;\ R_1 = 4$



Activity list SGS

Genotype: (<0,2,4,6,1,3,5,7],P)

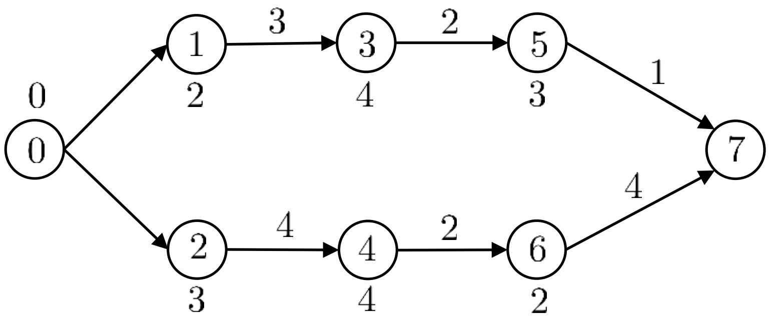
Father: (<0,1,3,5,2,4,6,7],S)

Mother: (<0,2,4,6,1,3,5,7],P)

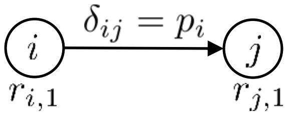
<u>Notation</u>:

SGS	Schedule Generation Scheme
P	Parallel
S	Serial

Genetic Algorithm: Two-Point Crossover



$|\mathcal{R}| = 1;\ R_1 = 4$



Random choice of $q_1 < q_2$
with $q_1, q_2 \in \{0, \dots, n+1\}$

<_,_,_|_,_|_,_,_]

Father: (<0,1,3|5,2|4,6,7],S)

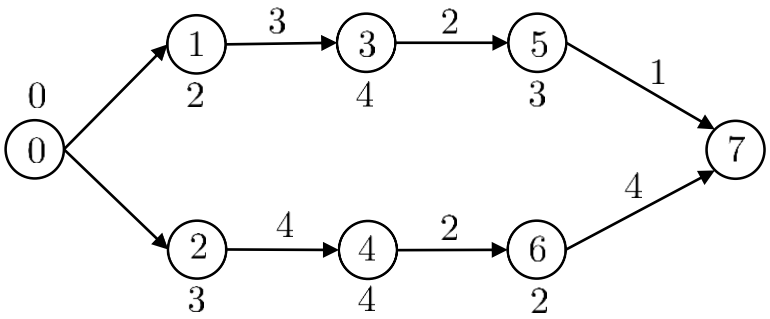
Mother: (<0,2,4,6,1,3,5,7],P)

The first q_1 activities are taken from the father

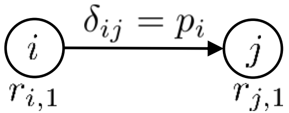
SGS from father

Son: (<0,1,3,_,_,_,_,_],S)

Genetic Algorithm: Two-Point Crossover



$| \mathcal{R} | = 1;\ R_1 = 4$



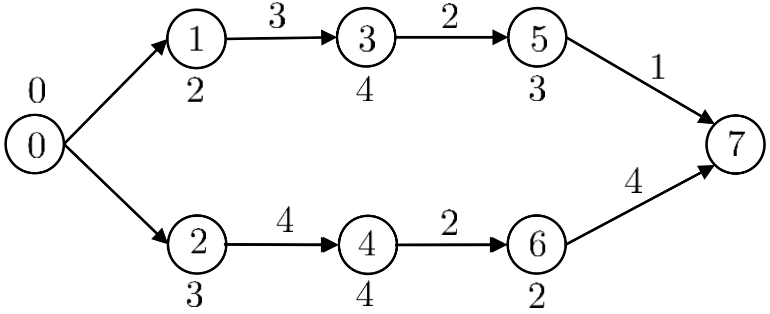
Father: (<0,1,3,5,2,4,6,7],S)

Mother: (<~~0~~,2,4,6,~~1~~,~~3~~,5,7],P)

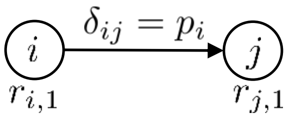
The next $q_1 - q_2$ activities are taken from the mother

Son: (<0,1,3,2,4,_,_,_],S)

Genetic Algorithm: Two-Point Crossover



$|\mathcal{R}| = 1;\ R_1 = 4$



Father: (<0,1,3,5,~~2~~,~~4~~,6,7],S)

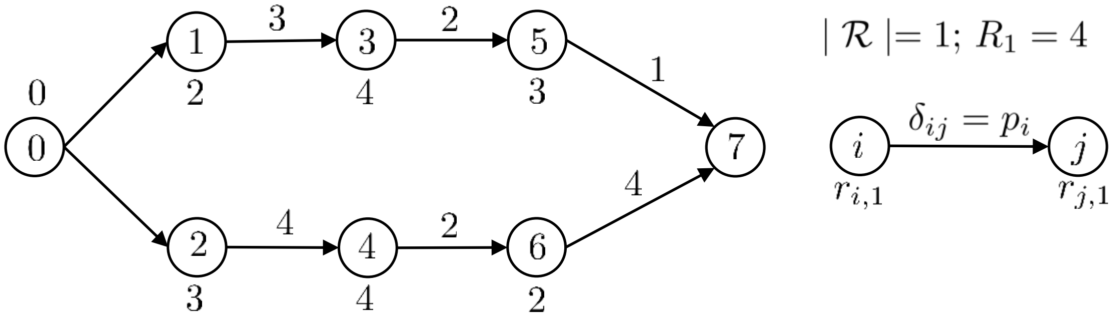
Mother: (<~~0~~,2,4,6,~~1~~,~~3~~,5,7],P)

Remaining activities from the father

Son: (<0,1,3,2,4,5,6,7],S)

Daughter: analogously

Genetic Algorithm: Mutation



Go through the activity list from position $p = 1$ to $n - 1$: For each position p with some probability (Hartmann uses 5%) exchange activity on position p with the one on position $p + 1, \dots, n$ if feasible w.r.t. precedence constraints. The selected position is random.

With some probability (Hartmann uses 5%) change the schedule generation scheme from “S” to “P” or vice versa.

Example: (<0,1,3,5,2,4,6,7],S)

Simulated Annealing for the RCPSP/M

Simulated Annealing (SA):

- First proposed in 1983 by Kirkpatrick et al.
- See Chapter 1 by Nikolaev and Jacobson in Gendreau and Potvin (2010)

SA-Application to the RCPSP: Bouleimen and Lecocq (2003)

References:

Bouleimen, K. and Lecocq, H. (2003): A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, European Journal of Operational Research, Vol. 149, pp. 268-281.

Kirkpatrick, S., Gelatt Jr., C. D. and Vecchi, M. P. (1983): "Optimization by Simulated Annealing“. Science Vol. 220 (4598), pp. 671–680.

Nikolaev, A.G. and Jacobson, S.H. (2010): Simulated Annealing, in: Gendreau M. and Potvin, J.-Y. (Editors): Handbook of Metaheuristics, 2nd Ed., Springer, New York.

Simulated Annealing: Notation and General Framework

Notation:

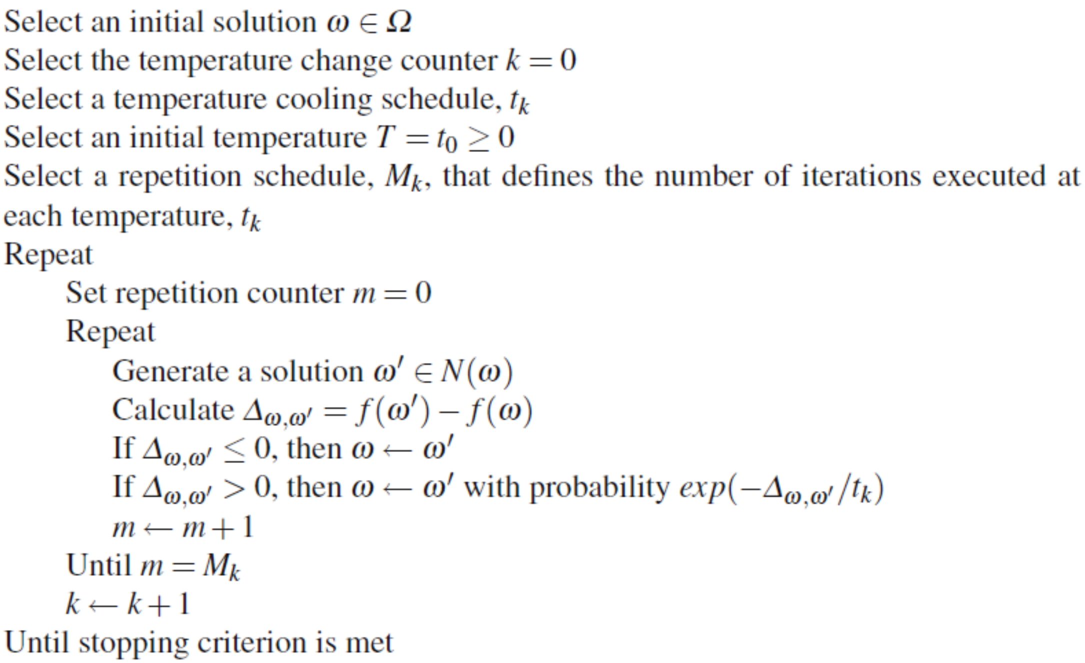
Ω: Solution space

$\omega \in \Omega$: Solution in the solution space

$f(\omega)$: Objective function of solution ω

$\omega' \in N(\omega)$: One solution in the neighborhood of solution ω

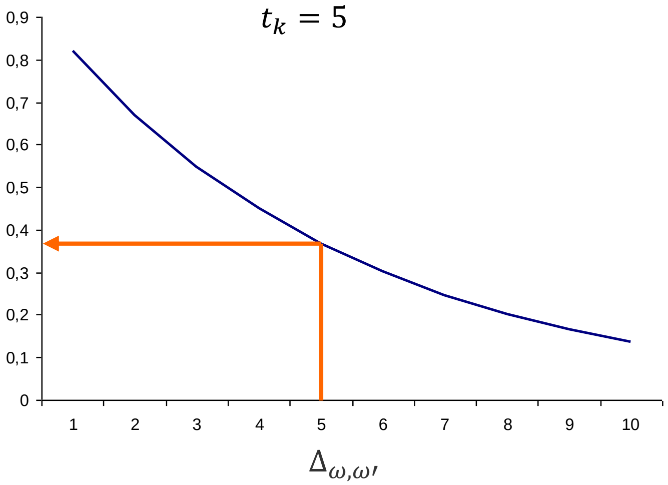
General Framework:

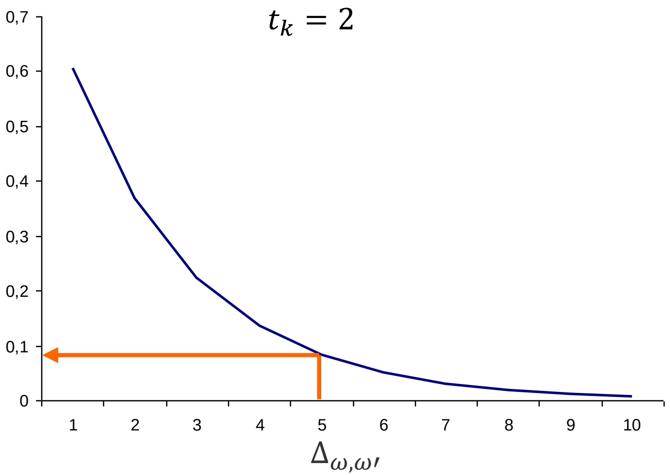


```
Select an initial solution $\omega \in \Omega$
Select the temperature change counter $k = 0$
Select a temperature cooling schedule, $t_k$
Select an initial temperature $T = t_0 \geq 0$
Select a repetition schedule, $M_k$, that defines the number of iterations executed at each temperature, $t_k$
Repeat
    Set repetition counter $m = 0$
    Repeat
        Generate a solution $\omega' \in N(\omega)$
        Calculate $\Delta_{\omega,\omega'} = f(\omega') - f(\omega)$
        If $\Delta_{\omega,\omega'} \leq 0$, then $\omega \leftarrow \omega'$
        If $\Delta_{\omega,\omega'} > 0$, then $\omega \leftarrow \omega'$ with probability $exp(-\Delta_{\omega,\omega'}/t_k)$
        $m \leftarrow m + 1$
    Until $m = M_k$
    $k \leftarrow k + 1$
Until stopping criterion is met
```

Acceptance Probability

$$Prob = \frac{1}{e^{\left(\frac{\Delta_{\omega,\omega'}}{t_k}\right)}}$$



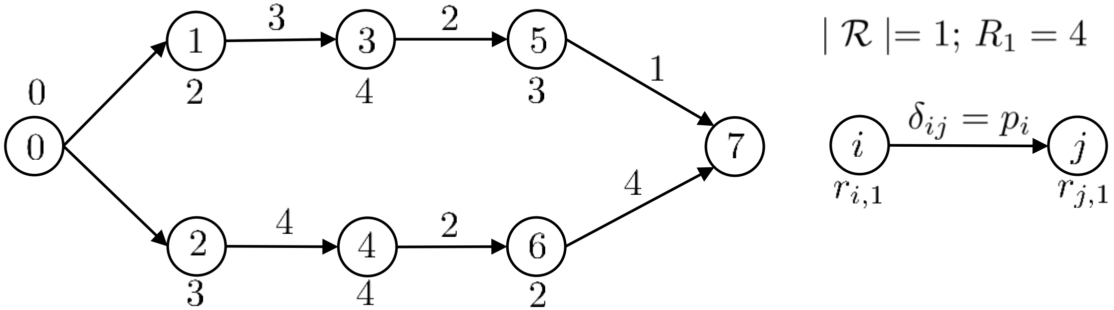


Observation: As the temperature t_k is lowered in the course of the SA-algorithm the acceptance probabilities of worse solutions ω' decreases.

Note: Independent of t_k we have: $\lim\limits_{\Delta_{\omega,\omega'\to\infty}} e^{-\Delta_{\omega,\omega'}/t_k} = 0$ and $\lim\limits_{\Delta_{\omega,\omega'\to 0}} e^{-\Delta_{\omega,\omega'}/t_k} = 1$

Solution Representation and Neighborhood

- Solution representation: Activity list
- Neighbor solution:
  - Selection of an activity (in the example activity 1)
  - Assign the selected activity to a random position within the allowed range (in the example [1,…,6])



Selected solution:

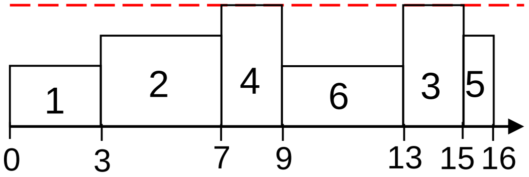
<0,1,2,4,6,3,5,7]

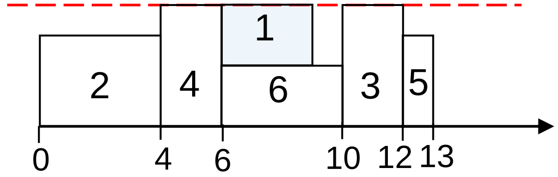
Neighborhood:

<0|1,2,4,6|3,5,7]

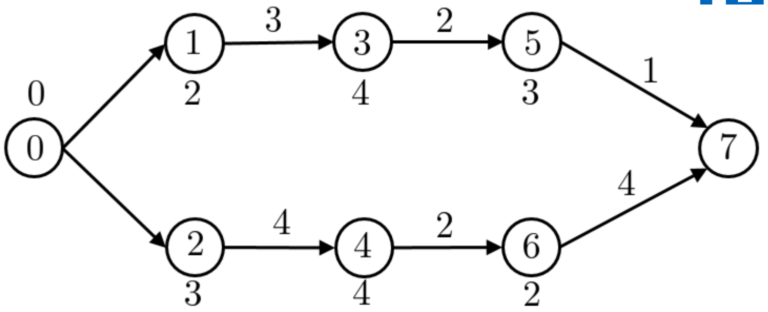
Neighborhood solution:

<0|2,4,1,6|3,5,7]





Cooling Schedule and Number of Iterations



Many different cooling schedules.

Geometric cooling schedule: $t_k = 0.9 \cdot t_{k-1}$

- Initial temperature t_0 = maximum difference between any two neighbor solutions
- Number of solutions at each temperature level = M_k (= size of the neighborhood).

Example: $t_k = 0.9 \cdot t_{k-1}$, $M_k = 5$ for all k, stopping criterion = 100 generated solutions,
initial temperature = upper bound (= $\sum_{i \in V} p_i = 16$) – lower bound ($ES_7 = 10$)= 6.

$k = 0, t_0 = 6$: Generate 5 solutions (total of 5 solutions).

$k = 1, t_1 = 5.4$: Generate 5 solutions (total of 10 solutions).

$k = 2, t_2 = 4.86$: Generate 5 solutions (total of 15 solutions).

…

$k = 19, t_{19} = 0.81$: Generate 5 solutions (total of 100 solutions)

Genetic Algorithm for the Resource Leveling Problem

Reference: Li, H., Xiong, L., Liu, Y. and Li, H. (2018): An effective genetic algorithm for the resource leveling problem with generalised precedence relations, in: International Journal of Production Research Vol. 56 (5) 2054-2075.

Problem addressed: Minimal time lags start-to-start with δ_{ij}. Objective function is the resource leveling problem (min sum weighted variance of the resource demand)

Solution Representation

$$\begin{pmatrix} RK \\ SK \end{pmatrix} = \begin{pmatrix} rk_0, rk_1, \ldots .rk_{n+1} \\ sk_0, sk_1, \ldots .sk_{n+1} \end{pmatrix}$$

$0 \leq rk_i \leq 1$ and $0 \leq sk_i \leq 1$

rk_i denotes the priority of activity i

sk_i denotes the start time of activity i as a convex combination between the dynamic earliest and latest start time.

Calculation of Dynamic Earliest and Latest Start Times

Let $V' \in V$ be a subset of activities which have already been scheduled, let S' be the partial schedule and let d_{ij} be the longest path between activity i and j in the network $N = (V, E, \delta)$. For each non-scheduled activity $i \ \in V \backslash V'$ dynamic earliest and latest start times are calculated according to

$$ES_i(S') = \max\{d_{0,i}, S_h + d_{hi} \,|\, h \in V' \text{and } (h, i) \in E\}$$

$$LS_i(S') = \min\{T^{\max} - d_{i,n+1}, S_j - d_{ij} |\, i \in V' \text{and } (i, j) \in E\}$$

Nochmal mit delta und d durchdenken, siehe auch Folie hinten.

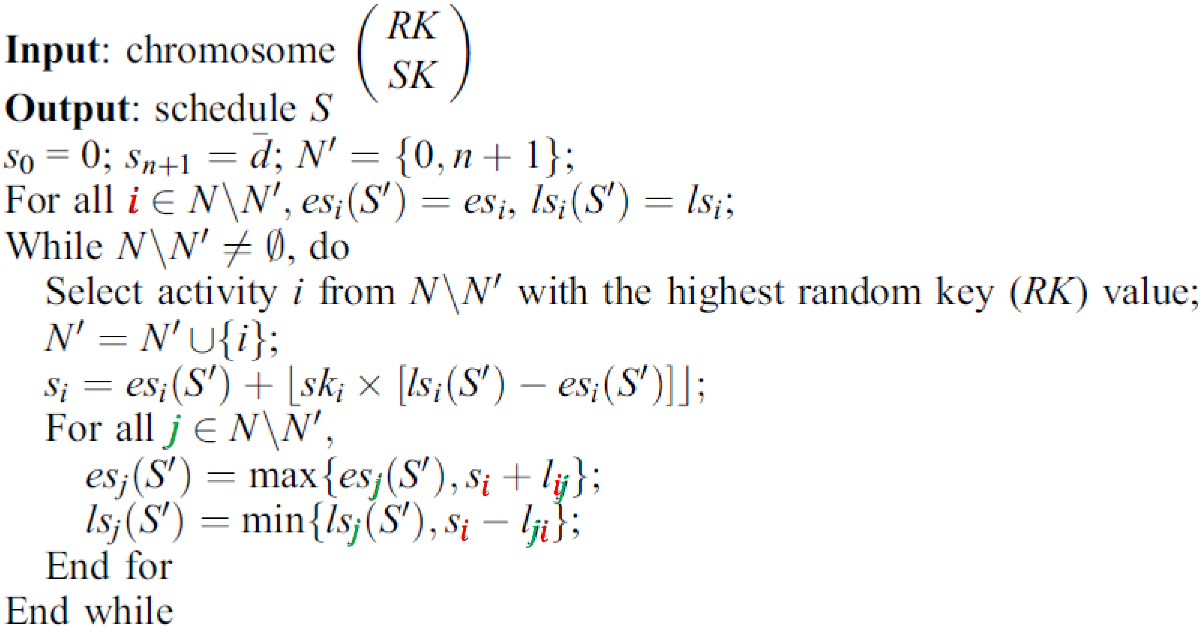
Mapping of a Chromosome into a Schedule

Notation of Li et al. (2018):

N= Set of activities (in our notation V)

$\bar{d}$ = Maximum process duration (in our notation $T^{\max}$)

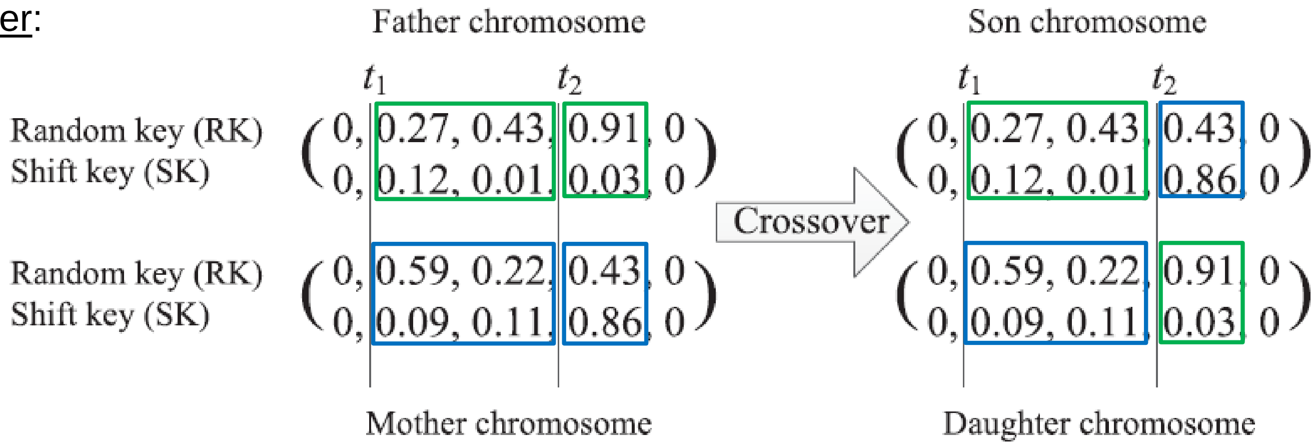
l_{ij}= longest path from node i to node j (start-start-relations, in our notation d_{ij})



```
Input: chromosome $\begin{pmatrix} RK \\ SK \end{pmatrix}$
Output: schedule $S$
$s_0 = 0;\ s_{n+1} = \bar{d};\ N' = \{0, n+1\};$
For all $i \in N \backslash N'$, $es_i(S') = es_i$, $ls_i(S') = ls_i$;
While $N \backslash N' \neq \emptyset$, do
    Select activity $i$ from $N \backslash N'$ with the highest random key ($RK$) value;
    $N' = N' \cup \{i\}$;
    $s_i = es_i(S') + \lfloor sk_i \times [ls_i(S') - es_i(S')] \rfloor$;
    For all $j \in N \backslash N'$,
        $es_j(S') = \max\{es_j(S'), s_i + l_{ij}\}$;
        $ls_j(S') = \min\{ls_j(S'), s_i - l_{ji}\}$;
    End for
End while
```

Crossover and Mutation

Two-Point Crossover:



Mutation:

```
For $i = 1, \ldots, n+1$
        if $RAN[0,1] < P_{Mutation}$ then
                $rk_i = RAN[0,1]$
                $sk_i = RAN[0,1]$).
```

Two-Phase Local Improvement Procedure

Phase 1: For a given schedule S order the activity according to ascending start times:

Select the next activity i:

Calculate the earliest start time of activity i given S:

$$ES_i(S) = \max\{S_h + \delta_{hi} \mid h \in V \text{and } (h, i) \in E\}$$

Calculate the latest start time of activity i given S:

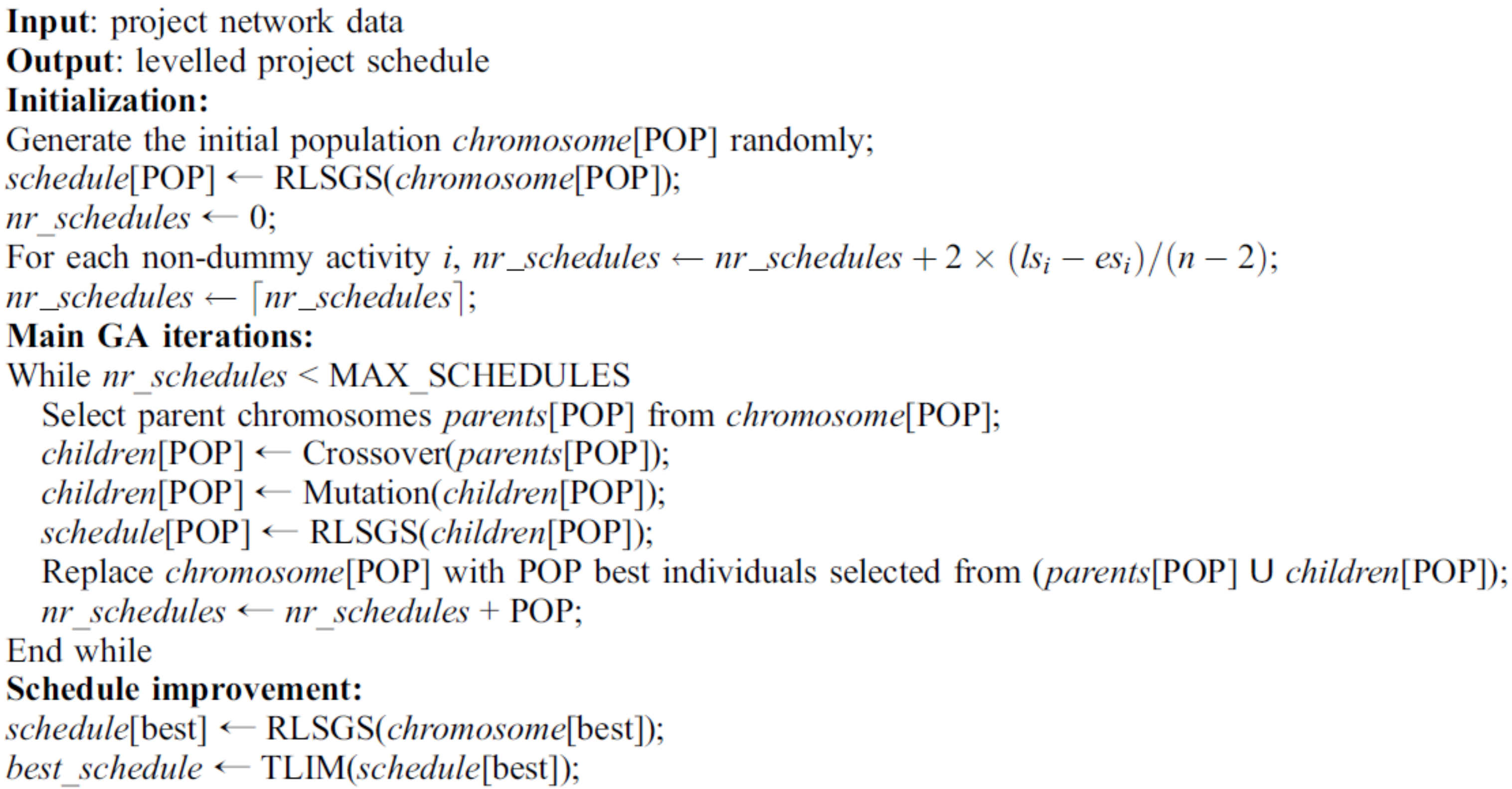
$$LS_i(S) = \min\{S_j - \delta_{ij} \mid i \in V \text{ and } (i, j) \in E\}$$

Set S_i' to time t, $ES_i(S) \leq t \leq LS_i(S)$, which minimizes the resource leveling objective function value. Set $S_i \coloneqq S_i'$

Phase 2: For a schedule S order the activity according to descending start times:

Apply the same procedure as for ascending start times.

Outline Genetic Algorithm



```
Input: project network data
Output: levelled project schedule
Initialization:
Generate the initial population chromosome[POP] randomly;
schedule[POP] ← RLSGS(chromosome[POP]);
nr_schedules ← 0;
For each non-dummy activity i, $nr\_schedules \leftarrow nr\_schedules + 2 \times (ls_i - es_i)/(n-2)$;
$nr\_schedules \leftarrow \lceil nr\_schedules \rceil$;
Main GA iterations:
While nr_schedules < MAX_SCHEDULES
    Select parent chromosomes parents[POP] from chromosome[POP];
    children[POP] ← Crossover(parents[POP]);
    children[POP] ← Mutation(children[POP]);
    schedule[POP] ← RLSGS(children[POP]);
    Replace chromosome[POP] with POP best individuals selected from (parents[POP] U children[POP]);
    nr_schedules ← nr_schedules + POP;
End while
Schedule improvement:
schedule[best] ← RLSGS(chromosome[best]);
best_schedule ← TLIM(schedule[best]);
```

Lecture 11: The Stochastic RCPSP

References:

- Ballestin, F. (2007): When it is worthwhile to work with the stochastic RCPSP? Journal of Scheduling, Vol. 10, pp. 153 – 166.
- Chen, F., Kolisch, R., Wang, L. and Mu, C. (2015): An estimation of distribution algorithm and new computational results for the stochastic resource-constrained project scheduling problem, Flexible Services and Manufacturing Journal, Vol. 27 (4), 585-605.
- Fliedner, T. and Kolisch, R. (2016): Performance and robustness of priority policies for static and dynamic multi-project scheduling under uncertainty, Working Paper.

Problem Setting

- As in the (deterministic) RCPSP except that the duration of activities are stochastic.
- It is assumed that the distribution of the duration of each activity is given by a discrete distribution.
- The objective is to minimize the expected makespan.

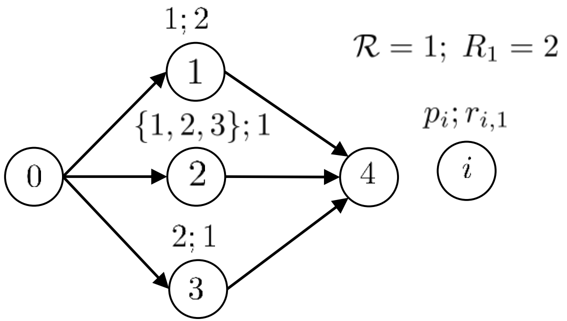
Scheduling Policies instead of Schedules

- Since the duration of an activity is not know before the activity is finished, a solution cannot be given by a vector of start and associated finish times.
- Instead, a solution is given by a scheduling policy Π, which defines for each time t and the information about the activities completed until t and in process at t which activities are started at t. Information about any finish time after t is not considered, as this information is not available at t (non-anticipativity constraint).
- For a given realization of the activity durations, a policy Π maps the activity durations p into a schedule of start (and finish) times $S(p)$: Π: $\mathbb{R}^{+}_{n+2} \rightarrow \mathbb{R}^{+}_{n+2}$.
- Let denote p a vector with realizations of activity durations, then $S^{\Pi}(p)$ is the schedule obtained by policy Π and the associated makespan is $S^{\Pi}_{n+1}(p)$.
- The objective of the stochastic RCPSP is to minimize the expected makespan $E(S^{\Pi}_{n+1}(p))$.

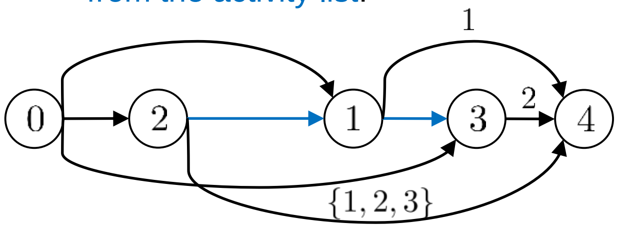
Activity-Based Policies

- Activity-based policies are also called stochastic serial schedule generation schemes (see Ballestin 2007).
- A policy Π is defined by an activity list and the stochastic version of the serial schedule generation scheme.
- The activity list has to consider the precedence constraints, i.e., in the list an activity cannot be before any of its successors.
- The schedule generation scheme is an adapted version of the serial schedule generation scheme: For a given sample of activity durations p, activities are scheduled with the serial schedule generation scheme using the additional constraint that $S_i \leq S_j$ if i is before j in the list L. Obviously, this is the introduction of a start-to-start precedence constraints with weight $\delta_{i,j} = 0$.

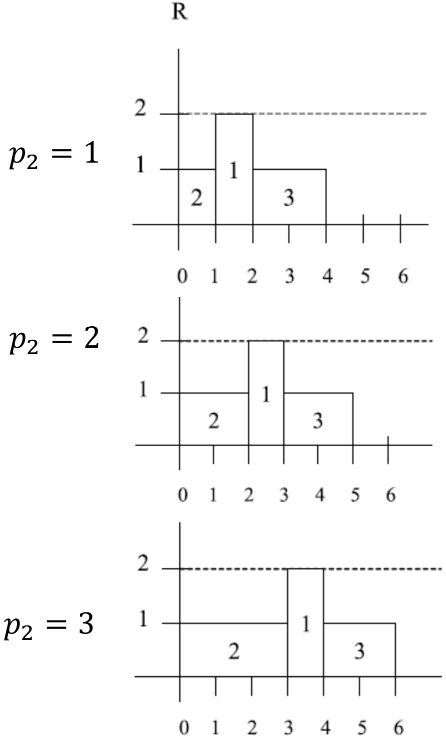
Example: Activity-based Policies



Activity graph after inserting the precedence constraints resulting from the activity list:



$L =< 0,2,1,3,4]$



When scheduling activity 3, it is not known how long the processing time of activity 2 will be. In case of $p_2 = 1$, starting activity 3 at $t = 0$, would delay the start time of activity 1, which is in the list before activity 3 and thus has priority over activity 3.

$E(S^{\Pi}_{n+1}(p))$=5

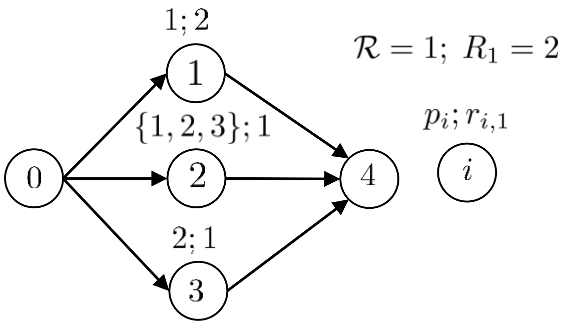
For two samples $(p_1, \dots, p_i, \dots, p_n)$ and $(p'_1, \dots, p'_i, \dots, p'_n)$ with $p_j = p'_j$ for $j \neq i$ and $p_i \leq p'_i$ it holds $E(S^{\Pi}_{n+1}(p)) \leq E(S^{\Pi}_{n+1}(p'))$

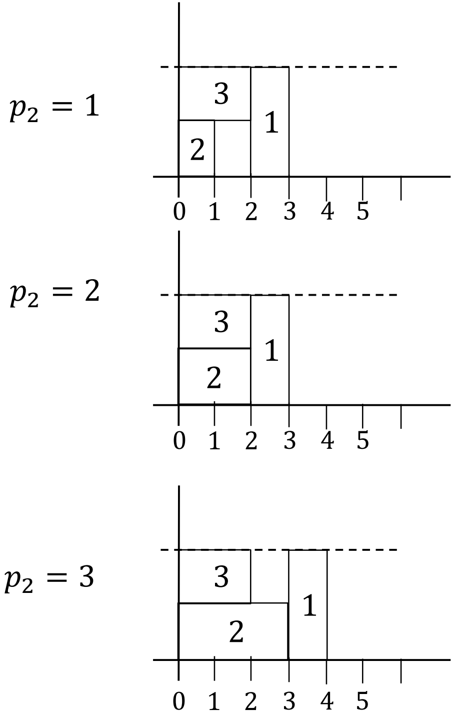
Resource-based Policies

- Resource-based policies are also called stochastic parallel schedule generation schemes (see Ballestin 2007).
- The activity list has to consider the precedence constraints, i.e., in the list an activity cannot be before any of its successors.
- The schedule generation scheme is the parallel schedule generation scheme: For a given sample of activity durations p, activities are scheduled with the parallel schedule generation scheme.

Example: Resource-based Policies

$L =< 2,1,3]$

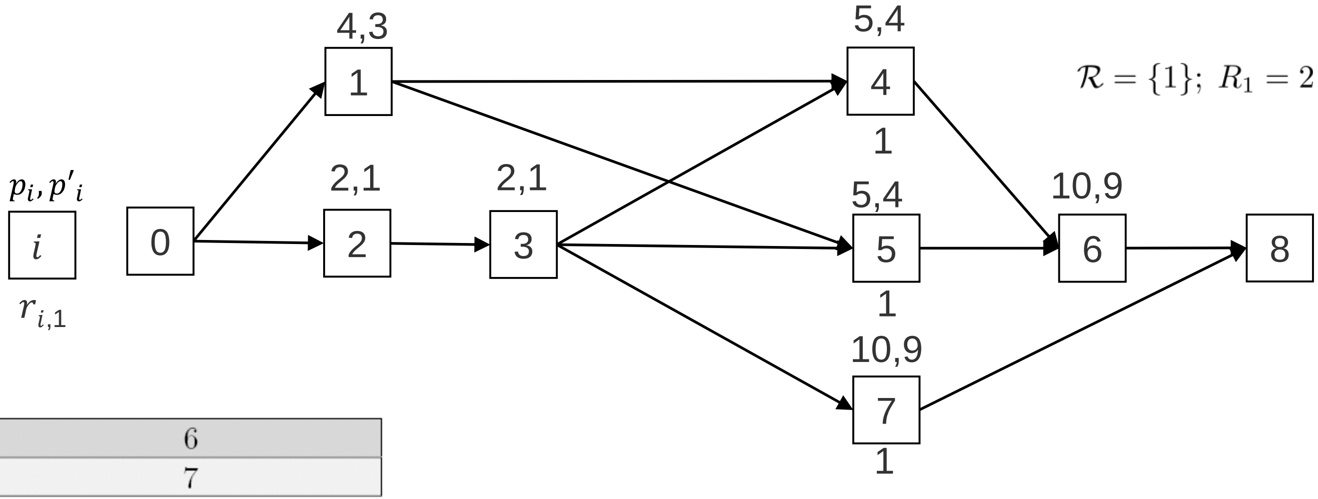




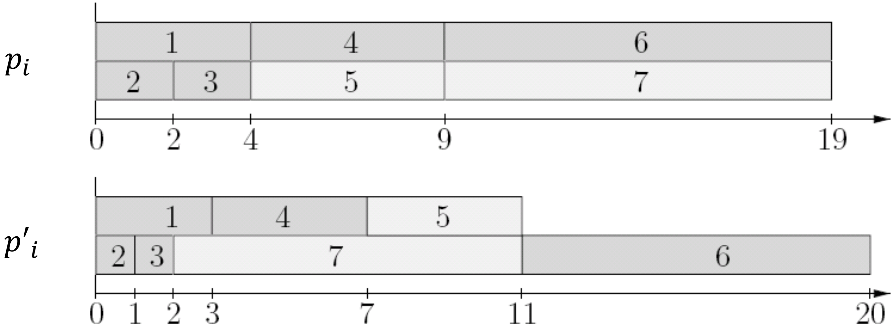
When scheduling activity 3, it is not known how long the processing time of activity 2 will be. In case of $p_2 = 1$, starting activity 3 at $t = 0$, would delay the start time of activity 1, which is in the list before activity 3 and thus has priority over activity 3.

$E(S_{n+1}^{\Pi}(p))$=3.33

Graham Anomaly

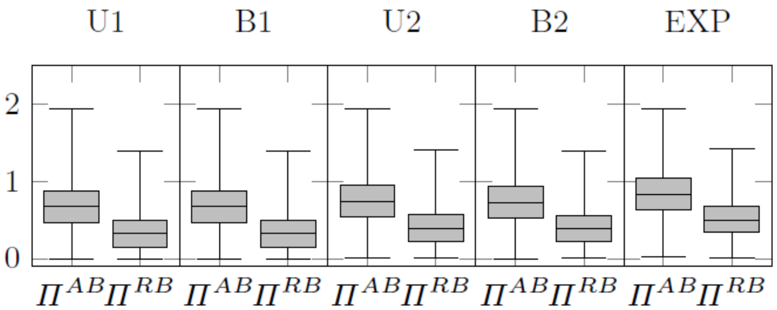


$\mathcal{R} = \{1\};\ R_1 = 2$



$L =< 0,1,2,3,4,5,6,7,8]$

Comparison of Activity- and Resource-Based Policies



Boxes = 25% and 75% percentile
Whiskers = min and max value

Distribution for all feasible lists.

- U1 / B1 = Uniform / Beta distribution with variance $E(p)/3$
- U2 / B2 = Uniform / Beta distribution with variance $(E(p))^2/3$
- EXP = Exponential distribution with variance $(E(p))^2$

Takeaway: The expected project duration of resource-based policies is much smaller than that of activity-based policies.

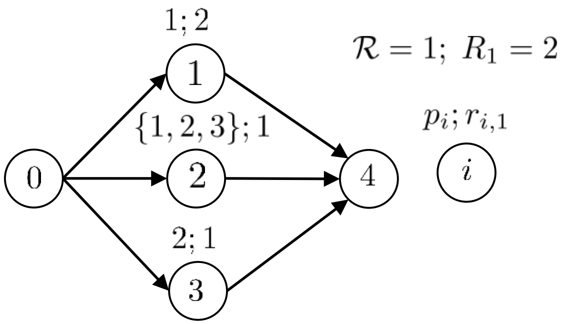
Reference: Fliedner and Kolisch (2016)

Heuristics for the SRCPSP: Preliminaries

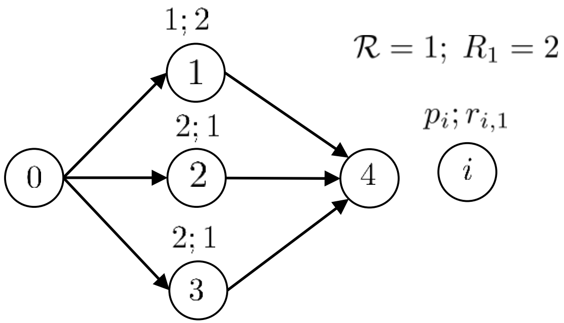
Deterministic project = Project where for each activity the expected duration is used.

Example:

Stochastic project



Deterministic project

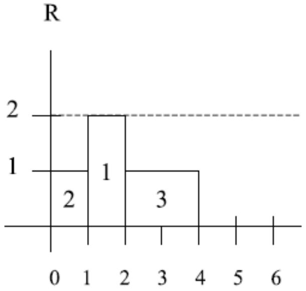


Heuristics for the RCPSP: Preliminaries

Mapping of a deterministic schedule S into an activity list L: $S \rightarrow L$. Sort the activities according to ascending start times (see Forward-Backward improvement, slide 137 et seqq.)

Example:

Schedule S



Activity list L

$L = < 2{,}1{,}3]$

Heuristics

Priority rule LFT

1. Use priority rule LFT and the serial SGS to generate a list L for the deterministic project.
2. Derive a schedule S from L by using the serial or the parallel SGS. Based on schedule S generate list L: $S \rightarrow L$.
3. Policy Π is defined by list L and the adapted serial scheduling generation scheme.
   For each sample of the project, generate a schedule S with policy Π. The quality of the policy is assessed based on the expected makespan.

Regret-based biased random sampling (for RCPSP see slide 136)

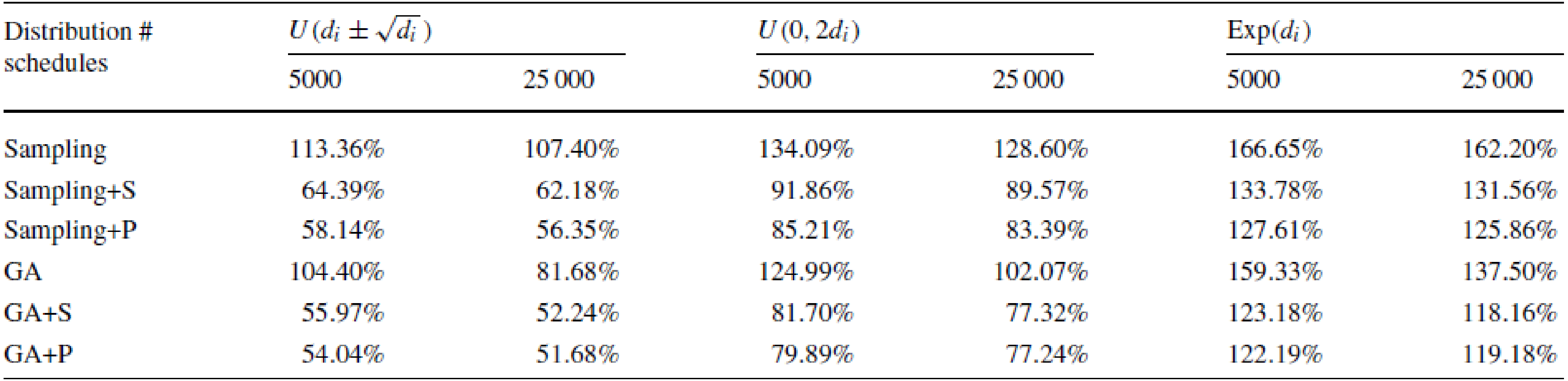
1. Use regret-based random sampling and the serial SGS to generate a list L for the deterministic project.
2. Derive a schedule S from L by using the serial or the parallel SGS. Based on schedule S generate list L: $S \rightarrow L$.
3. Policy Π is defined by list L and the adapted serial scheduling generation scheme.
   For each sample of the project, generate a schedule S with policy Π. The quality of the policy is assessed based on the expected makespan

Heuristics

Genetic algorithm of Hartmann

1. Generate a start population using regret-based random sampling.
2. Double the size of the population by generating offspring lists .
3. For each element (list) in the population, Derive a schedule S from L by using the serial or the parallel SGS. Based on schedule S generate list L: $S \rightarrow L$.
4. Policy Π is defined by list L and the adapted serial scheduling generation scheme.
   For each sample of the project, generate a schedule S with policy Π. The quality of the policy is assessed based on the expected makespan

Heuristics: Performance



Distribution # schedules	$U(d_i \pm \sqrt{d_i})$		$U(0, 2d_i)$		$\mathrm{Exp}(d_i)$	
	5000	25 000	5000	25 000	5000	25 000
Sampling	113.36%	107.40%	134.09%	128.60%	166.65%	162.20%
Sampling+S	64.39%	62.18%	91.86%	89.57%	133.78%	131.56%
Sampling+P	58.14%	56.35%	85.21%	83.39%	127.61%	125.86%
GA	104.40%	81.68%	124.99%	102.07%	159.33%	137.50%
GA+S	55.97%	52.24%	81.70%	77.32%	123.18%	118.16%
GA+P	54.04%	51.68%	79.89%	77.24%	122.19%	119.18%

Source: Table 1 in Ballestin (2007).

- 5,000 and 25,000: Number of schedules generated.
- „+S/+P“ = Generating the list L by use of the for the serial/parallel SGS
- „Sampling“ = regret-based biased random sampling
- For all algorithms, a sample with 100 scenarios has been used.
- Performance measure = percentage increase above the resource-unconstrained duration of the deterministic project.

Takeaways:

- Adding “+S/+P”, improves results considerably. “+P” is slightly better than “+S”.
- Genetic algorithm slightly better than regret-based biased random sampling.
- Increase of variance of activity durations decreases performance.