2. Project management

- In what follows, we consider production processes where only a single item of a specific product is produced in the planning horizon
- In this case specific instruments for planning and controlling are necessary to enable an efficient reliable execution
- The production of the single item itself can be interpreted as a project to be managed

2. Outline of the chapter

1. Basic definitions
2. Analysis of the project structure
3. Time analysis and planning
4. Analysis of the time table flexibility

2.1 Basic definitions

Definition 2.1.1 (Project – according to DIN)

According to DIN 69901, a **project** is

"an endeavor characterized entirely and essentially by

- specific terms, for example its objective, temporal, human or other resource limitations
- differences from other endeavors
- a project-specific organization"

Project model in this course

Definition 2.1.2 (Project – in the course)

As a **project** we define a finite set of tasks (procedures) whose processing consumes a predefined period of time and whose execution has to fulfill predefined time schedule restrictions. These restrictions can consist in specific minimal and maximal temporal distances of the beginning of tasks. Additionally, the total execution time of the project is restricted.

Project schedule I

1. Description of the project
   - Informal and formal definition of the total project
   - Computation of the available resources
   - Assignment of competences
   - Integration into the existing organization
2. Analysis of the project
   1. Structure analysis
      - Task Decomposition of the total project
      - Determination of the existing precedence restrictions between the different tasks

Project schedule II

2. Time analysis
   - Determination of the task processing times
   - Determination of the existing time restrictions between the different tasks
3. Analysis of the capacity and material requirements
4. Cost analysis

3. Planning of the project structure

   Mapping of the total project in a model, e.g.,

   - Network plan
   - Linear program

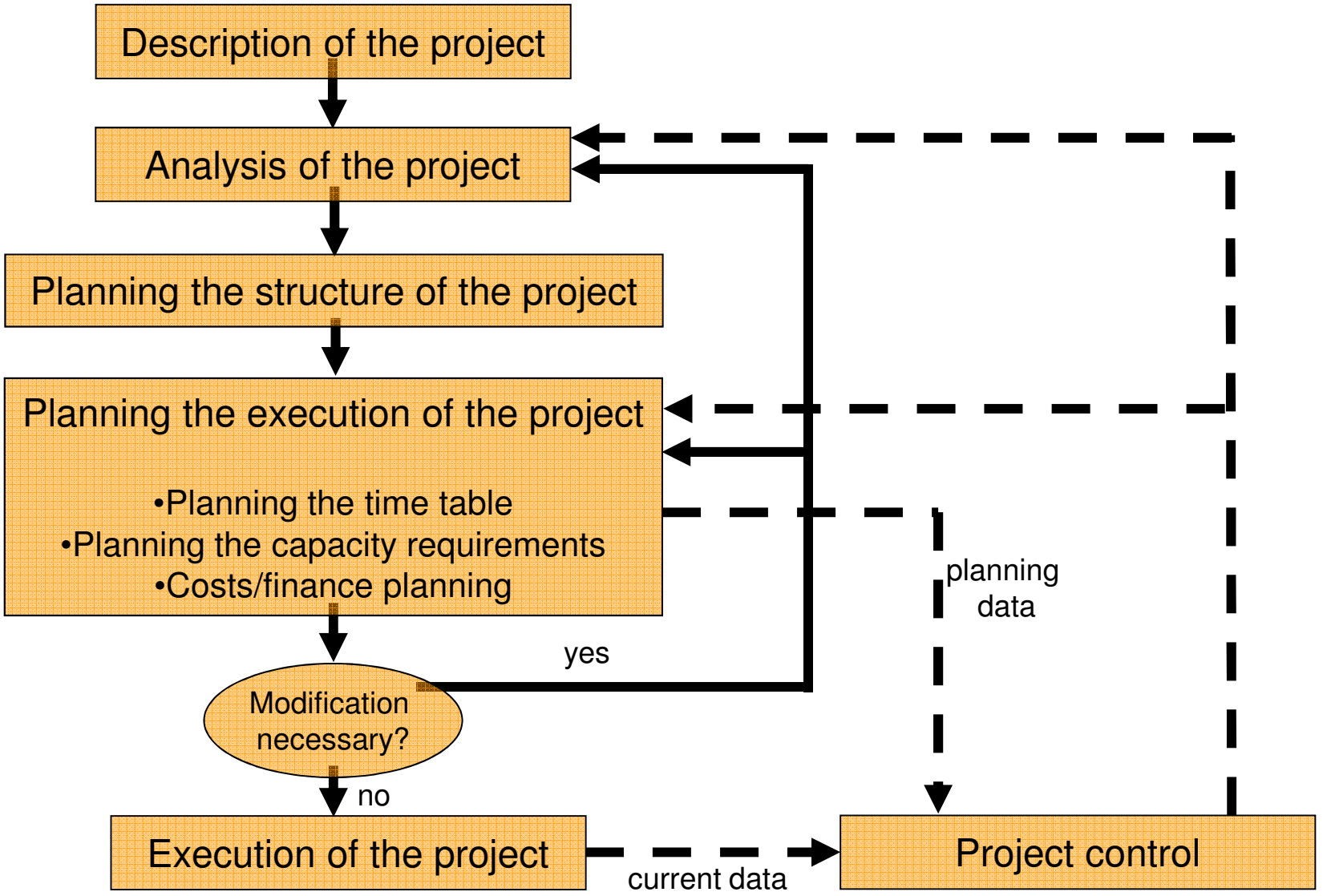
Project schedule III

4. Time planning
   1. Determination of feasible time tables
      - Determination of the earliest possible starting and ending positions of tasks
      - Determination of the latest possible starting and ending positions of tasks
      - Determination of the shortest possible project duration
   2. Analysis of the flexibility of the given time table
      - Determination of the range of alternative starting positions of each task
      - Estimating the consequences of exceeding the existing time windows for each task

Project schedule IV

5. Planning the capacity and material requirements
   - Optimal use of the existing capacities
6. Planning the resulting costs
7. Control of the project execution
   1. Controlling the consequences of costs
   2. Controlling the time tables
   3. Controlling the execution of the tasks

Conducting a project



2.2 Analysis of the project structure

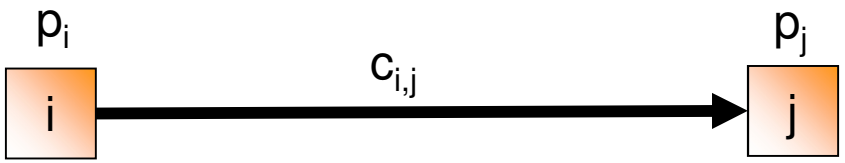
Definition 2.2.1 (Project task network)

A **project task network** *V* is a connected network with *N+2* nodes *{0,1,…,N,N+1}* and without cycles of positive length with two additional marked nodes *0* and *N+1* and the following interpretation:

- 0 is the exclusive source of the network symbolizing the beginning of the total project
- *N+1* is the exclusive sink of the network symbolizing the end of the total project
- Each node *i (1≤i≤N)* represents a task with a predefined processing time p_i
- Each edge *(i,j)* in the network has a weight $c_{i,j}$ that is interpreted as follows:
  - $c_{i,j} \geq 0$: The task j can begin at least $c_{i,j}$ time units after the beginning of task *i* (minimum distance restriction)
  - $c_{i,j} < 0$: The task i must begin at most $-c_{i,j}$ time units after the beginning of task j (maximum distance restriction)

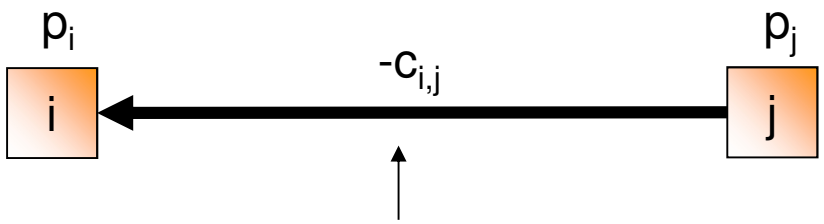
Distance requirements

- Minimum distance restriction



The beginning of task *j* has to respect a distance of at least $c_{i,j}$ time units to the beginning of task *i.*
p_i denotes the processing time of task i

- Maximum distance restriction



The beginning of task *i* has to respect a maximum distance of at most $-c_{i,j}$ time units to the beginning of task *j*

Attention: Changed orientation

Convention

Notation:

In each time table the beginning of task *i* is denoted by t_i

Generally, the project start is set to 0: $t_0 = 0$.

Sought:

Feasible time tables

Corollary 2.2.2 :

Let *V* be a project task network with an edge *(i,j)* with weight $c_{i,j}$. For every time table $t=(t_0,t_1,\ldots,t_{N+1})$ that fulfills the restrictions of definition 2.2.1 it holds:

$$t_i+c_{i,j} \leq t_j.$$

Proof of corollary 2.2.2

Case 1: $c_{i,j} \geq 0$

In this case task j begins at least $c_{i,j}$ after the beginning of task i.

$\Rightarrow t_j \geq t_i + c_{i,j}$

Case 2: $c_{i,j} < 0$

In this case task i must begin at most $-c_{i,j}$ after the beginning of task j.

$\Rightarrow t_i \leq t_j + \left(-c_{i,j}\right) \Leftrightarrow t_i \leq t_j + \left(-c_{i,j}\right) \Leftrightarrow t_i \leq t_j - c_{i,j}$

$\Leftrightarrow t_i + c_{i,j} \leq t_j \Leftrightarrow t_j \geq t_i + c_{i,j}$

Consequences

- Independent of its algebraic sign, distance requirements can be treated in the same way. Therefore, **maximum and minimum distance restrictions are equivalent**. This simplifies the following computations significantly.

Feasible time tables

Definition 2.2.3

Let $V = (E,\Gamma,c)$ be a project task network, with $E = \{0,1,...,N,N+1\}$. Additionally, let T be a predetermined time bound for the maximum project duration. A time table $t = (t_0, t_1 ..., t_N, t_{N+1})$ is **feasible** if and only if

$$t \in X_V^T = X = \left\{ \begin{array}{l} t \in IR_{\geq 0}^{N+2} \ / \\ \forall i \in \{0,1,...,N\} : \forall j \in \Gamma(i) : t_j \geq t_i + c_{i,j} \wedge \\ \forall i \in \{0,1,...,N+1\} : t_i \geq 0 \wedge t_{N+1} \leq T \end{array} \right\}$$

Note that X_V^T $(or\ X)$ defines the **set of feasible time tables**

Time-feasible region

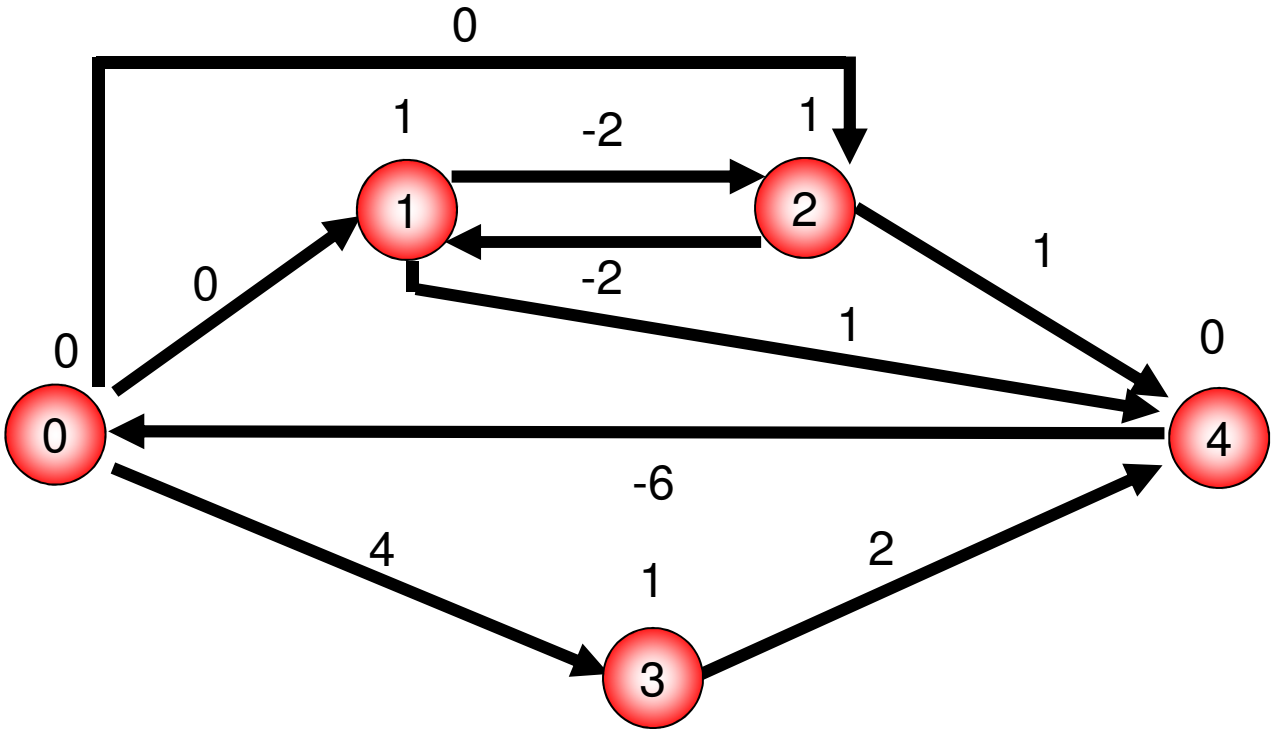
Observation 2.2.4

X represents a convex polyphedron which is called the **time - feasible region**.
It comprises all feasible time tables.

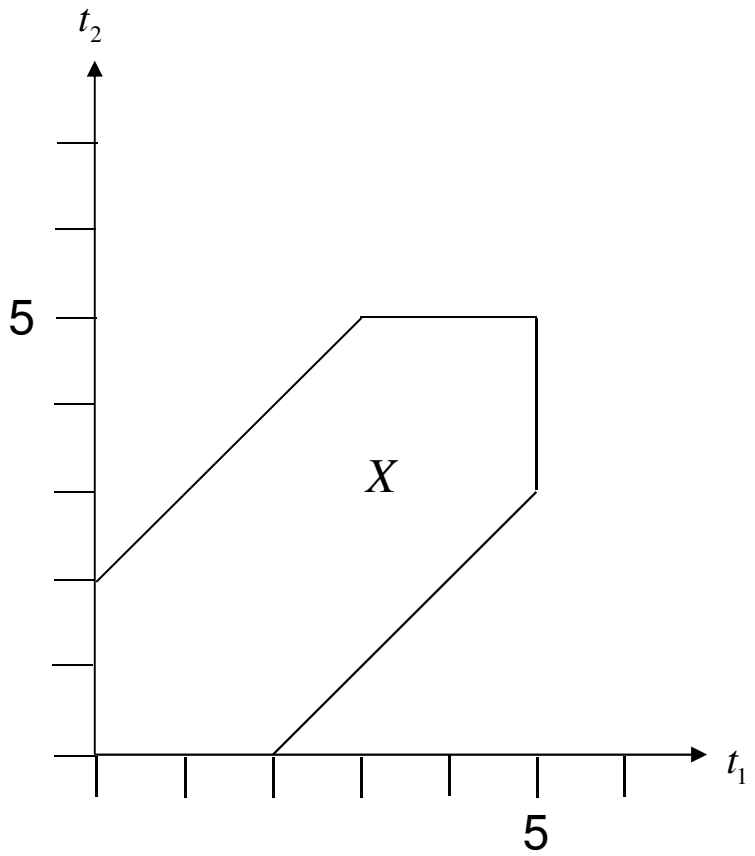
The time-feasible region is a multi-dimensional object that is difficult to visualize. But at the special case, that exactly two tasks *i* and *j* are not fixed, X can be projected into the *i*, *j*-plane.

Thus, this projection is called the t_i - t_j - *cut of* X.

Example: Project Network



Example: t_1 - t_2 cut of X :



Conclusion

Corollary 2.2.5 :

The defined time restrictions in a project task network are transitive requirements, i.e., a valid time table fulfilling existing time restrictions defined by the arcs $(i,j,c_{i,j})$ and $(j,k,c_{j,k})$ fulfills also the time restriction defined by the arc $(i,k,c_{i,j}+c_{j,k})$.

The **proof** is trivial.

Note that if there is a path p between two nodes i and j with length $l_{i,j}(p)$, each feasible time table has to fulfill the restriction: $t_j \geq t_i + l_{i,j}(p)$. Also note that the longest path of a node i to itself is defined as $l(i,i) = 0$.

2.3 Time analysis and planning

Generation of feasible time tables

We define:

- EB_j: Earliest possible beginning of task j in a feasible time table with respect to the project start which is set to the earliest beginning 0.
- LB_j: Latest possible beginning of task j in a feasible time table with respect to the project start and the restriction according to a maximum project duration.
- EE_j: Earliest possible end of task j in a feasible time table with respect to the project start which is set to the earliest beginning 0.
- LE_j: Latest possible end of task j in a feasible time table with respect to the project start and the restriction according to a maximum project duration.

Computing EB values

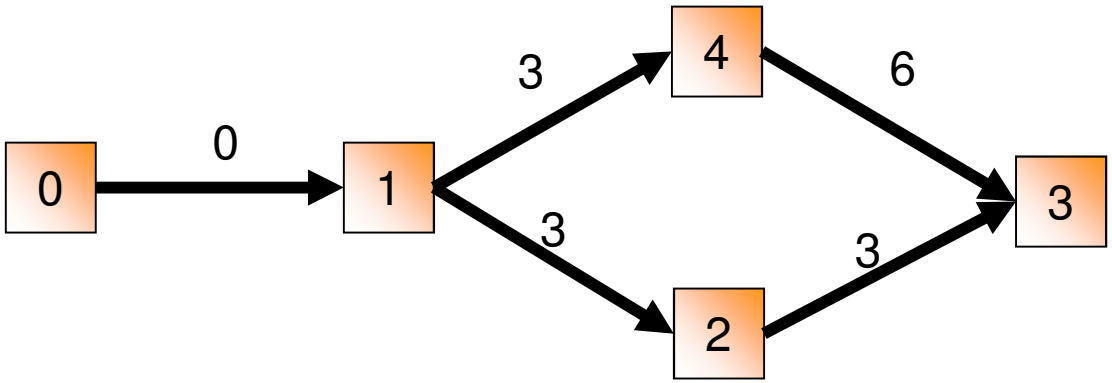
Lemma 2.3.1

It holds: $\forall i \in \{0,...,N+1\} : EB_i = l(0,i)$ while $\forall i, j \in \{0,...,N+1\} : l(i,j)$ defines the length of the longest path from node i to node j. Additionally, it holds: $EE_i = EB_i + p_i$.

Proof:

Assume t_i is the beginning of task i while all predecessors of i belong to the set $\Gamma^{-1}(i)$. In order to fulfill the existing time restrictions of the project, it holds for every path p with length $l_p^{j,i}$ starting at node j and ending at i: $t_i \geq t_j + l_p^{j,i}$. Since node 0 is the start of the total project and t_0 the respective point of time, all paths from 0 to i define minimum distances according to the earliest beginning of task i. Therefore, by fulfilling the requirement of the longest path, all such restrictions are obeyed.

Example



$t_0 = 0$
$EB_3 = ?$

It holds:

$t_1 \geq t_0 + 0$
$t_4 \geq t_1 + 3$
$t_2 \geq t_1 + 3$
$t_3 \geq t_4 + 6 \geq t_1 + 3 + 6 \geq t_0 + 0 + 9 \geq 0 + 9 = 9 = l(0,3)$

Computing LB values

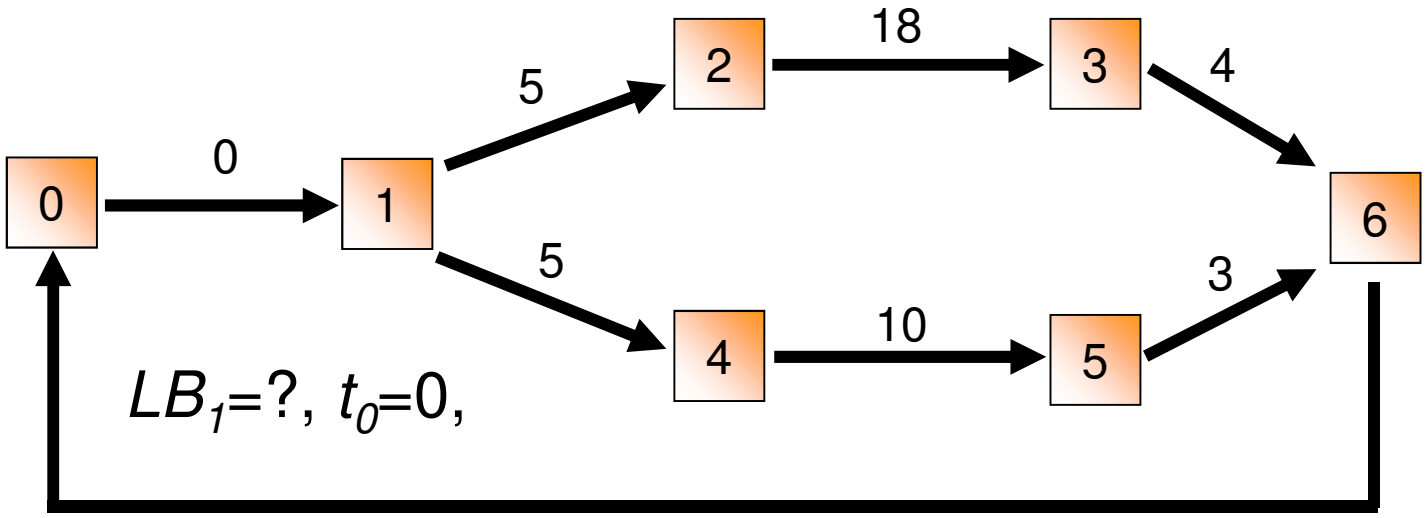
Lemma 2.3.2

If a project network gets extended by an arc $\langle N+1, 0, -T \rangle$, *it holds* : $\forall i \in \{0,\ldots,N+1\} : LB_i = -l(i,0)$ *while* $\forall i, j \in \{0,\ldots,N+1\}$: $l(i,j)$ *defines the length of the longest path from node* i *to node* j. *Additionally, it holds* : $LE_i = LB_i + p_i$.

Proof:

Is left as an assignment

Example



We give t_6 the largest feasible value t_6=56

Therefore, we get:

$t_3 \le t_6$-4

$t_5 \le t_6$-3

$t_4 \le t_5$-10$\le t_6$-13 and $t_2 \le t_3$-18$\le t_6$-22=34

$t_1 \le t_2$-5$\le t_6$-18 and $t_1 \le t_4$-5$\le t_6$-27=29

i.e., $t_1 \le 29 = LB_1$

Conclusion

- In order to compute the values *EB*, *LB*, *EE*, *LE* for each task, we need an **instrument to calculate** the **longest paths** from an arbitrary node to the sink as well as from the source to an arbitrary node in the network
- Such an instrument is known as the so-called "**Bellman-Ford algorithm**" that works tree-oriented to generate one or all the longest paths starting from a predefined node to all other nodes in the network

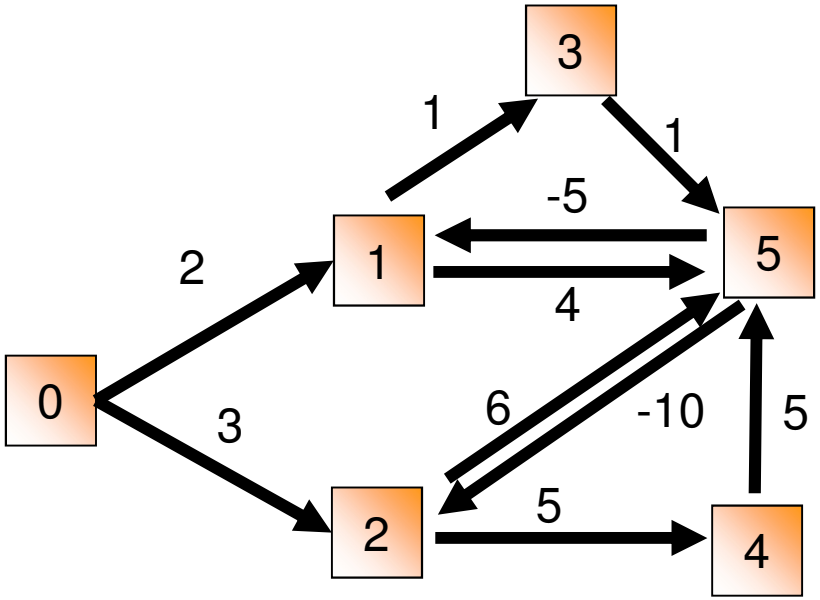
Bellman-Ford algorithm

Input: Network V=(E,Γ,c) and node $i_0 \in E$

1. Let $l(i_0)=0$ and iteration counter $r=0$. Additionally, B_0 is the tree comprising only node i_0 as its root
2. If all leafs of B_r are labeled or have no successor in V, the algorithm stops. Otherwise, continue with step 3
3. Starting with an unlabeled leaf i, consider all $j \in \Gamma(i)$
   1. j not in B_r: Insert j with $l(j)=l(i)+c_{i,j}$
   2. j already in B_r and $l(i)+c_{i,j}>l(j)$: Insert j with the larger path length and label the old one with all successors
   3. j already in B_r and $l(i)+c_{i,j}<l(j)$: Insert j as labeled leaf
   4. j already in B_r and $l(i)+c_{i,j}=l(j)$: Insert j with $l(j)$ if and only if j does not lay on the given way from i_0 to i
4. Increase r by 1; $B_{r+1}=B_r \cup$ {new nodes in last round}
5. Go to step 2

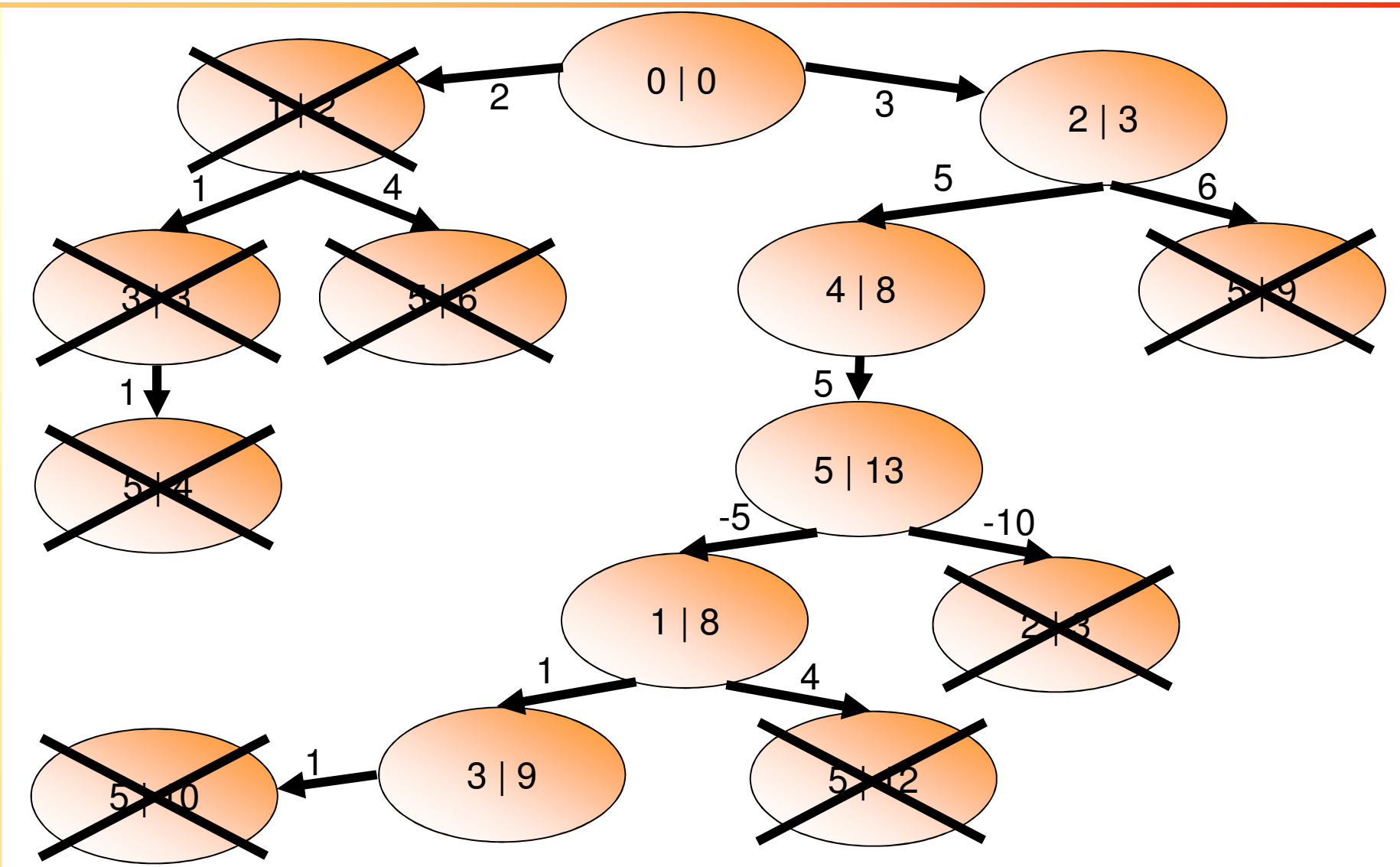
The algorithm has a time complexity of $O(|E|*|\Gamma|)$.

Example



Compute all the longest paths starting from node 0 to all other nodes

Bellmann-Ford algorithm – Calculation process

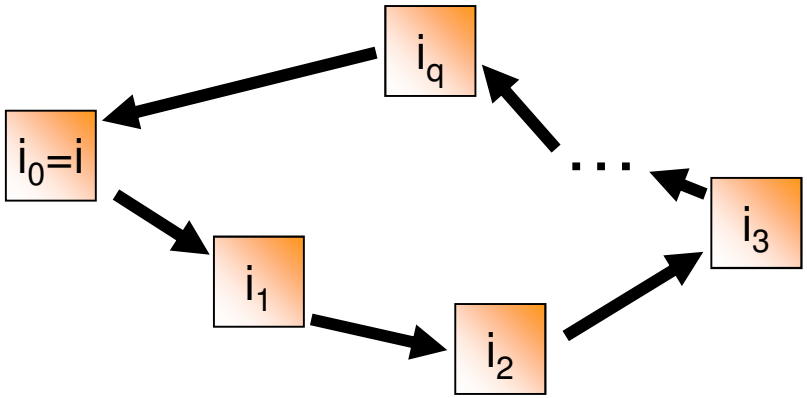


Results

Node i	$l(1,i)=EB_i$	Comment
0	0	
1	8	By visiting node 5
2	3	
3	9	
4	8	
5	13	

Cycles of positive length

- Are not allowed in project task networks
- The question is “Why?”
- Let us assume there is a cycle of positive length in the network



Cycle c with $l(c)>0$

What follows? $t_i+l(c)\leq t_i$ => This makes no sense !

Linear program for computing EB

$$Minimize \sum_{i=0}^{N+1} t_i$$

$$with:$$

$$\forall i \in \{0,\ldots,N\}: \forall j \in \Gamma(i): t_i + c_{i,j} \leq t_j$$

$$\forall i \in \{0,\ldots,N+1\}: t_i \geq 0$$

Linear program for computing LB

$$Maximize \sum_{i=0}^{N+1} t_i$$

with:

$$\forall i \in \{0,\ldots,N\}: \forall j \in \Gamma(i): t_i + c_{i,j} \leq t_j$$

$$\forall i \in \{0,\ldots,N+1\}: t_i + p_i \leq T$$

$$\forall i \in \{0,\ldots,N+1\}: t_i \geq 0$$

Longest paths

- We can state that the longest path from node 0 to node *N+1* determines the shortest possible execution duration of the total project
- Therefore, we define all tasks belonging to this path as **time-critical according to the shortest possible project execution duration**

2.4 Analysis of the time table flexibility

- In what follows, we compute time windows for a later or earlier beginning of each task assuming a given time table
- These time windows elucidate the flexibility of the considered time table with respect of occurring disturbances during its realization
- The sizes of these time windows are often calculated by **specific buffer times**

General time-table-oriented buffer times

- A fixed feasible time table $t \in X$ is predetermined
- Possible local relocations are considered
- We distinguish movements of the current task *i* beginning in direction of the project end from movements to the earliest position
- The first case is analyzed by the $TFB_i(t)$
- The second case is analyzed by the $TBB_i(t)$
- Changes of the positions of all other tasks by these movements of the beginning of task *i* are not allowed

$TFB_i(t)$ (Total Forward Buffer)

$$TFB_i(t) = t_i^l - t_i$$

while t_i^l defines according to plan t, the latest possible beginning of task i

i.e., it holds: $\left(t_0,\ldots,t_{i-1},t_i^l,t_{i+1},\ldots,t_{N+1}\right) \in X$

$$\Rightarrow t_i^l = \min\left\{T - p_i, t_j - c_{i,j} \mid j \in \Gamma(i)\right\}$$

$TBB_i(t)$ (Total Backward Buffer)

$$TBB_i(t) = t_i - t_i^e$$

while t_i^e defines according to plan t, the earliest possible beginning of task i

i.e., it holds: $\left(t_0,\ldots,t_{i-1},t_i^e,t_{i+1},\ldots,t_{N+1}\right) \in X$

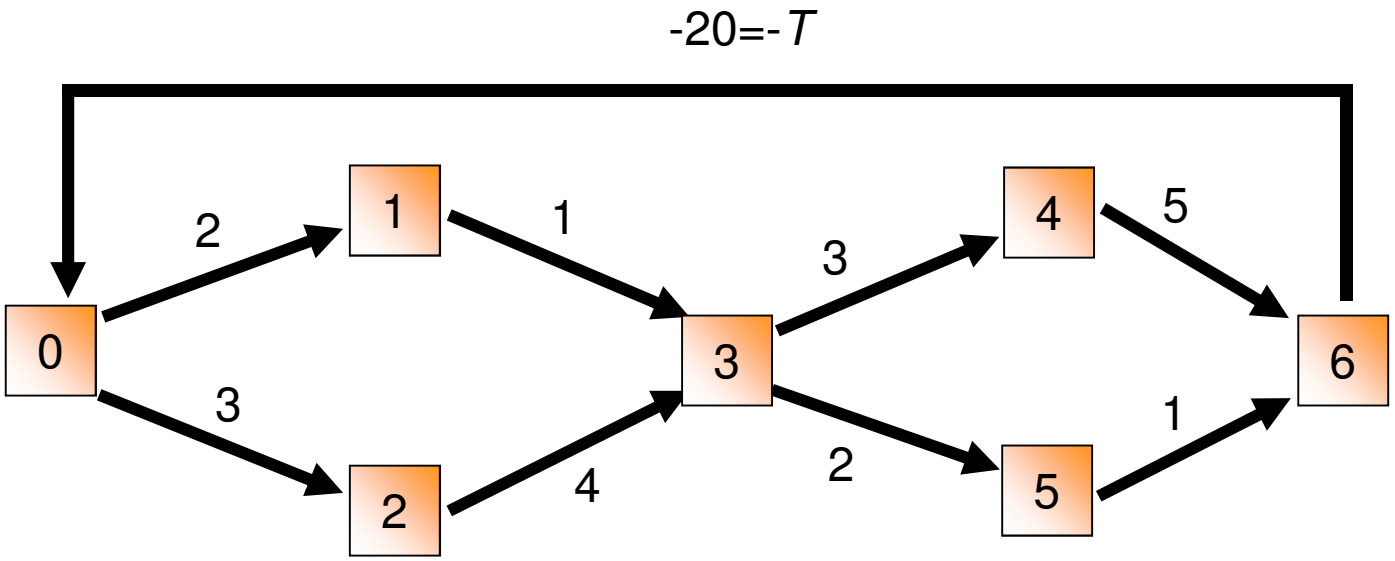
$$\Rightarrow t_i^e = \max\left\{0, t_j + c_{j,i} \mid j \in \Gamma^{-1}(i)\right\}$$

Total time-table-oriented buffer time

- The total general time-table-oriented buffer time ($TB_i(t)$) is defined as follows:

$$TB_i(t) = TFB_i(t) + TBB_i(t)$$

Example



Given time table:

Node *I*	0	1	2	3	4	5	6
t_i	0	4	5	10	15	16	20

$TFB_3(t)$, $TBB_3(t)$, $TB_3(t)$

- $TFB_3(t)=\min\{t_4-3,t_5-2,20\}-t_3=\min\{12,14,20\}-t_3=2$
- $TBB_3(t)= t_3-\max\{t_1+1,t_2+4,0\}=10-\max\{5,9,0\}=1$
- $TB_3(t)= TFB_3(t)+TBB_3(t)=2+1=3$

Extreme buffer times

- While the time-table-oriented buffer times are computed according to the definitions of a given feasible time table, the extreme buffer times generate available time windows for each task in the best and in the worst case
- Therefore, extreme buffer times illustrate independently of a given time table which tasks have to be scheduled carefully to generate a reliable execution of the total project. We focus on the following topics:
  - Worst case consideration: Calculation of the time windows of each task in an adversarial situation defined by a specific time table $t_{worstcase} \in X$
  - Best case consideration: Calculation of the time windows of each task in an fortunate situation defined by a specific time table $t_{bestcase} \in X$
  - Therefore, all considered plans have to be feasible and belong to X

Minimal buffer time ($MinB_i$)

- Here we ask for the worst case, i.e., the minimum time window available for each task independent of a given time table
- By inserting the edge ($N+1$,0,$-T$), we integrate the restriction of the maximum project duration of T time units into the project task network
- Therefore, $MinB_i$ is defined as follows:

$$MinB_i = \min \{ TB_i(t) \mid t \in X \}$$

Calculating the minimal buffer time ($MinB_i$)

Lemma 2.4.1

The minimum buffer time $MinB_i$ may be computed by:

$$MinB_i = \min\left\{l(h, j) - c_{h,i} - c_{i,j} \mid h \in \Gamma^{-1}(i) \wedge j \in \Gamma(i)\right\}$$

Proof:

The time window of each task i depends on the existing constellation of the respective predecessors and successors.

Therefore, we have to consider $\Gamma(i)$ and $\Gamma^{-1}(i)$.

Owing to the requirements of a feasible time table for each constellation $(h, j) \in \Gamma^{-1}(i) \times \Gamma(i)$ it holds:

$t_h + l(h, j) \leq t_j$.

Proof (continued):

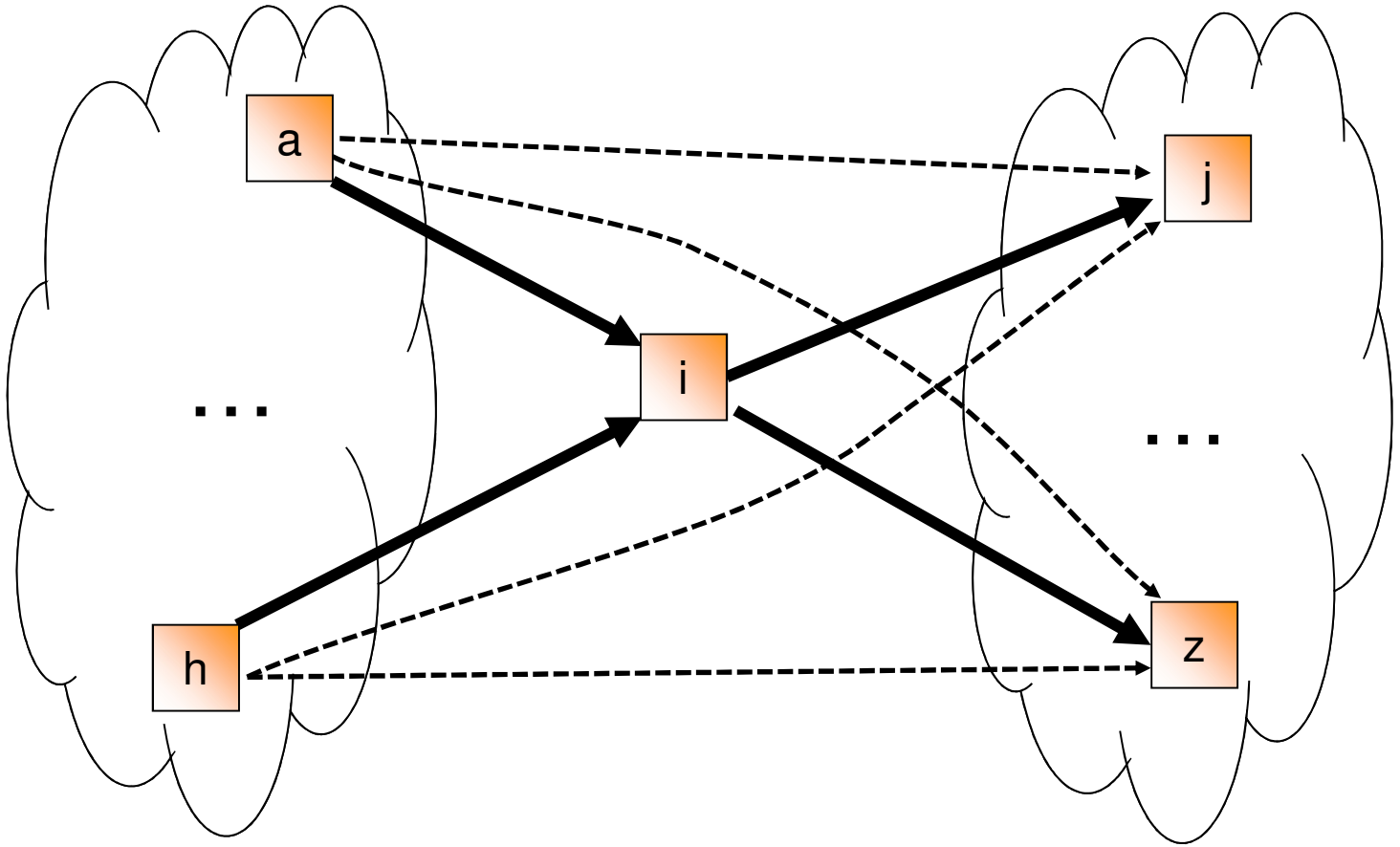
Therefore, each feasible time table t has to guarantee this minimum distance restriction between every constellation of predecessor and successor of i.

This leads to the following lower bound for each time table $t \in X$:

$$TB_i(t) \geq \min\left\{ l(h,j) - c_{h,i} - c_{i,j} \mid h \in \Gamma^{-1}(i) \wedge j \in \Gamma^{-1}(i) \right\}.$$

Therefore, we can conclude the defined calculation for $MinB_i$.

Illustration



Longest path between the respective tasks in the network

Maximal buffer time ($MaxB_i$)

- Here we ask for the best case, i.e., the maximum time window available for each task independent of a given time table
- By inserting the edge ($N+1$,0,$-T$), we integrate the restriction of the maximum project duration of T time units into the project task network
- Therefore, $MaxB_i$ is defined as follows:

$$MaxB_i = \max \{ TB_i(t) \mid t \in X \}$$

Calculation of $MaxB_i$

Lemma 2.4.2

The maximum buffer time $MaxB_i$ can be computed by:

$$MaxB_i = -\max\left\{l(j,h) + c_{h,i} + c_{i,j} \mid h \in \Gamma^{-1}(i) \wedge j \in \Gamma(i)\right\}$$

Proof:

Now we search for the constellation of t $\in$ X where i has the largest time window to be moved. Therefore, we have to ask for the maximum distance between all constellations of respective predecessors and successors. Consequently, we have to consider again $\Gamma(i)$ and $\Gamma^{-1}(i)$.

Proof (continued):

Note that due to the attributes of a project task network it holds:

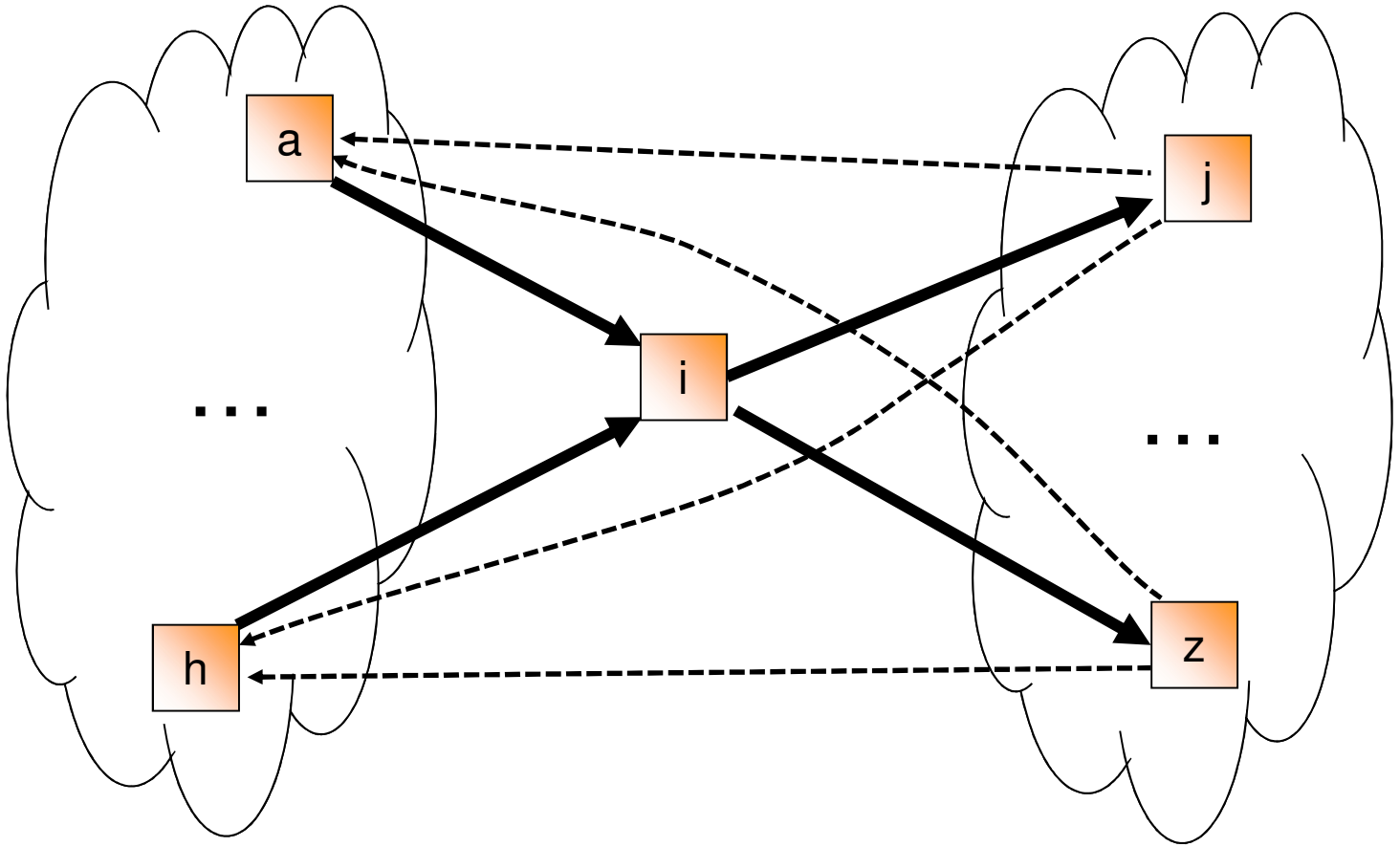
$l(j,h)+c_{i,j}+c_{h,i} \leq 0$. Therefore, we can compute the length of the maximum available time window by

$$\mathrm{TB}_{\mathrm{i}}(t) \leq -\max\left\{l(j,h)+c_{i,j}+c_{h,i} \mid h \in \Gamma^{-1}(i) \wedge j \in \Gamma(i)\right\}.$$

$\Rightarrow$ Consequently, we can derive the defined upper bound.

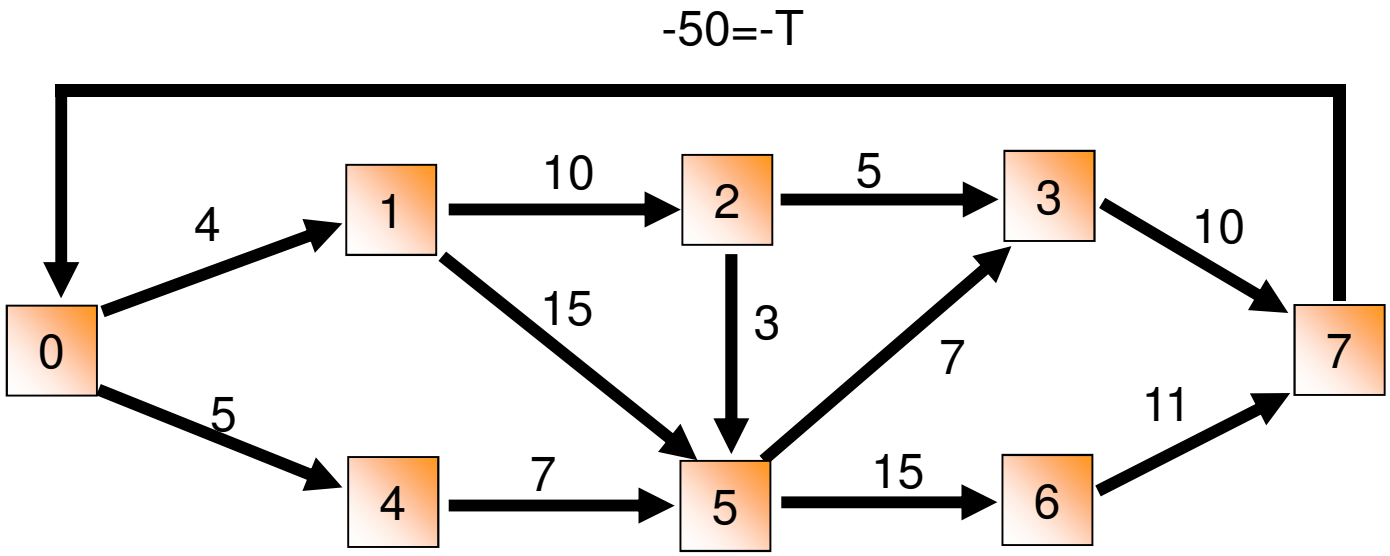
Owing to the requirements of a feasible time table for each constellation $(h, j) \in \Gamma^{-1}(i) \times \Gamma(i)$ it holds: $t_j + l(j,h) \leq t_h$ and, as a result, the maximum distance between these two tasks is restricted by $t_j - t_h \leq -l(j,h)$. That means we have to search the most restrictive maximum distance restriction between a successor and a predecessor of i.

Illustration



Longest path between the respective tasks in the network

Example



Sought: $MinB_2$ and $MaxB_2$

$$\Gamma^{-1}(2)=\{1\},\Gamma(2)=\{3,5\}$$

$$MinB_2=\min\{I(1,3)-15,I(1,5)-13\}=\min\{22-15,15-13\}=\min\{7,2\}=2$$

$$MaxB_2=-\max\{I(3,1)+15,I(5,1)+13\}=-\max\{-36+15,-20+13\}$$

$$=-\max\{-21,-7\}=7$$

Simple observations

- A small value for $MaxB_i$ underlines that the task i can be seen as a very critical process in the project whose scheduling must be handled carefully

- In contrast to this, a large value for $MinB_i$ can be seen as an indicator for an uncritical task

Attention

- In literature, **four additional buffer times**, which are introduced next, can be found
- But: Using these buffer times can lead to **inconsistent results**. Therefore, we do not concentrate our description to a pure definition but itemize additionally existing limitations for a possible application
- We characterize the following buffer times as **specific time-table-oriented buffer times**. This can be explained by the fact that their definition assumes the existence of an extreme constellation given by a specific time table. **Unfortunately, this assumed plan is not always feasible,** wherefore the computed times are not valid

Total buffer time TBT_i

- The TBT_i assumes the case where all successors of task *i* begin at their *LB* and all predecessors at their *EB*
- Computed by:

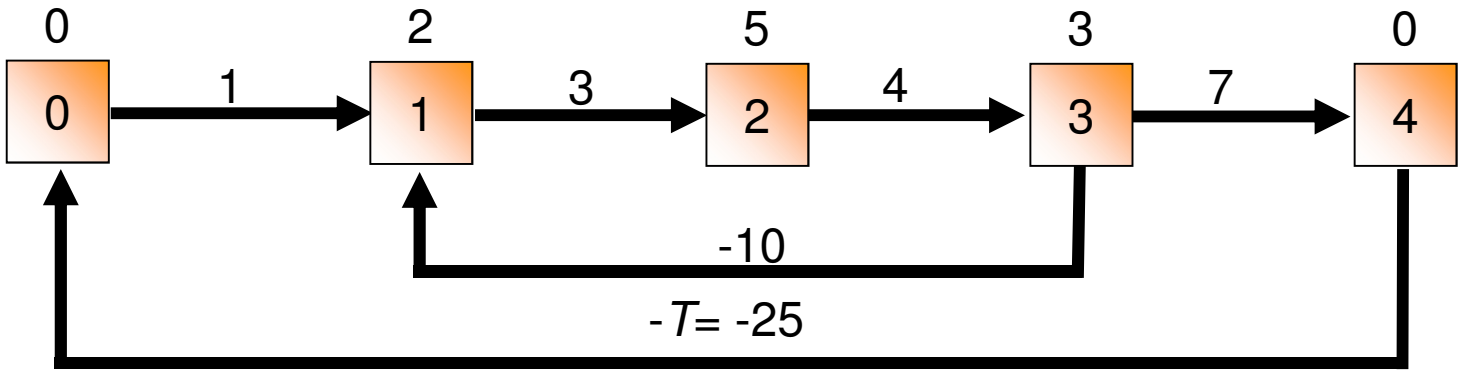
$$TBT_i = LB_i - EB_i$$

- The supposed extreme time table is **not always feasible**, i.e., the value of TBT_i is not always valid since it is not always possible to begin all predecessors at their *EB* and all successors at their *LB*. This plan can become infeasible

Problems with TBT

- Closer analyses of the calculation of TBT_i underline that the supposed constellation can lead to infeasible time tables if the project task network **comprises a cycle of negative length** after erasing the edge (*N*+1,0,-*T*)
- These cycles can define **restrictive local requirements** that are not respected by *LB* and *EB* and which are defined exclusively according to the longest path in the whole network. By neglecting these local requirements, the assumed time table can become infeasible

Example



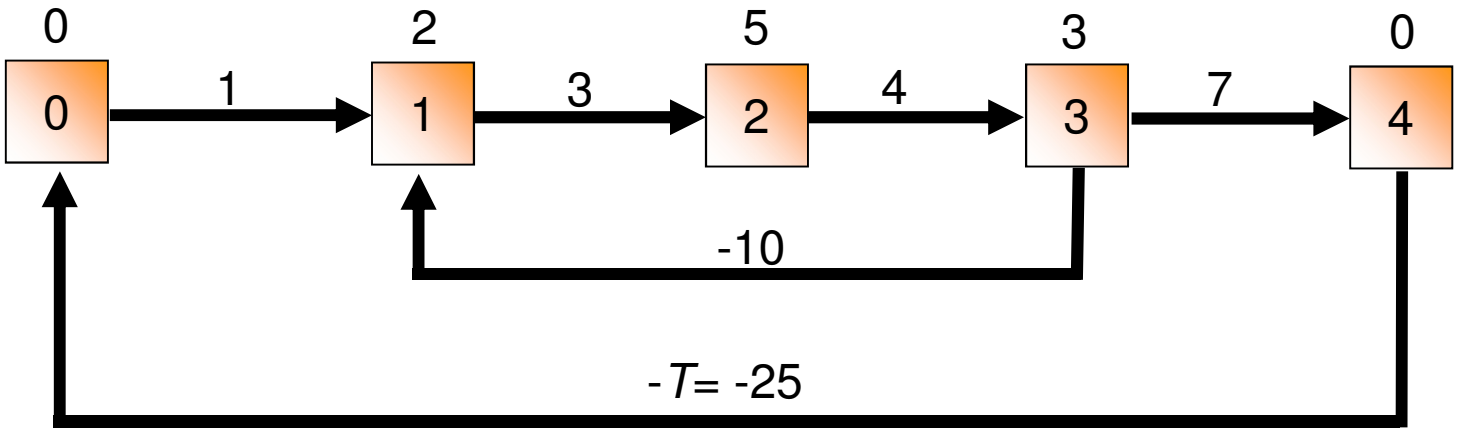
We compute TBT_2 by:

- EB_1=1, LB_1=11
- EB_2=4, LB_2=14, and therefore, TBT_2=14-4=10
- EB_3=8, LB_3=18

BUT:

The assumed time table with t_1=1 and t_3=18 is not feasible!
The maximum distance restriction (3,1,-10) is not fulfilled!

Using the maximum buffer time instead



We compute $MaxB_2$ by:

=-max{*I*(3,1)+4+3}=-max{max{-10,-25+8}+4+3}=-max{-10+7}
=3

The maximum distance restriction (3,1,-10) limits the available time window to 3 time units

TBT vs. MaxB

Lemma 2.4.3

If a project task network contains no cycle after erasing the edge $(N+1,0,-T)$, it holds: $\forall i \in \{0,1,...,N,N+1\}$:

$MaxB_i = TBT_i$

Proof:

We assume that after erasing the edge $(\mathrm{N}+1,0,-\mathrm{T})$ the considered project task network does not contain any cycle. We compute $MaxB_i$ as follows:

$$MaxB_i = -\max\left\{l(r,t) + c_{i,r} + c_{t,i} \mid t \in \Gamma^{-1}(i) \wedge r \in \Gamma(i)\right\}$$

The computation defined above computes the length of a cycle in the network. Owing to our assumptions, this cycle always contains the edge $(\mathrm{N}+1,0,-\mathrm{T})$. Therefore, in this case we can derive:

$$MaxB_i = -\left(l(i,N+1) + l(0,i) - T\right) = \underbrace{T - l(i,N+1)}_{LB_i} - \underbrace{l(0,i)}_{EB_i} = TBT_i$$

Free buffer time (FBT_i)

- The FBT_i assumes that all predecessors and successors of task *i* begin to their *EB*
- Always applicable
- It holds:

$$FBT_i = TFB_i(t_{EB})$$

with: t_{EB} defines the time table where each task i begins at $t_i = EB_i$

Free backward buffer time ($FBBT_i$)

- The $FBBT_i$ assumes that all predecessors and successors of task *i* begin to their *LB*
- Always applicable
- It holds:

$$FBBT_i = TBB_i(t_{LB})$$

with: t_{LB} defines the time table where each task *i* begins at $t_i = LB_i$

Independent buffer time (IBT_i)

- The IBT_i assumes that all predecessors of task i begin at *LB* while…
- …all successors begin at *EB*
- An infeasible constellation is frequently defined by this parameter setting
- Therefore, for task *i* the so-called "independent interval" is generated as follows:

$$\forall i \in \{0, 1, ..., N, N+1\}: IPI_i = \left[t_i^l, t_i^u\right]$$

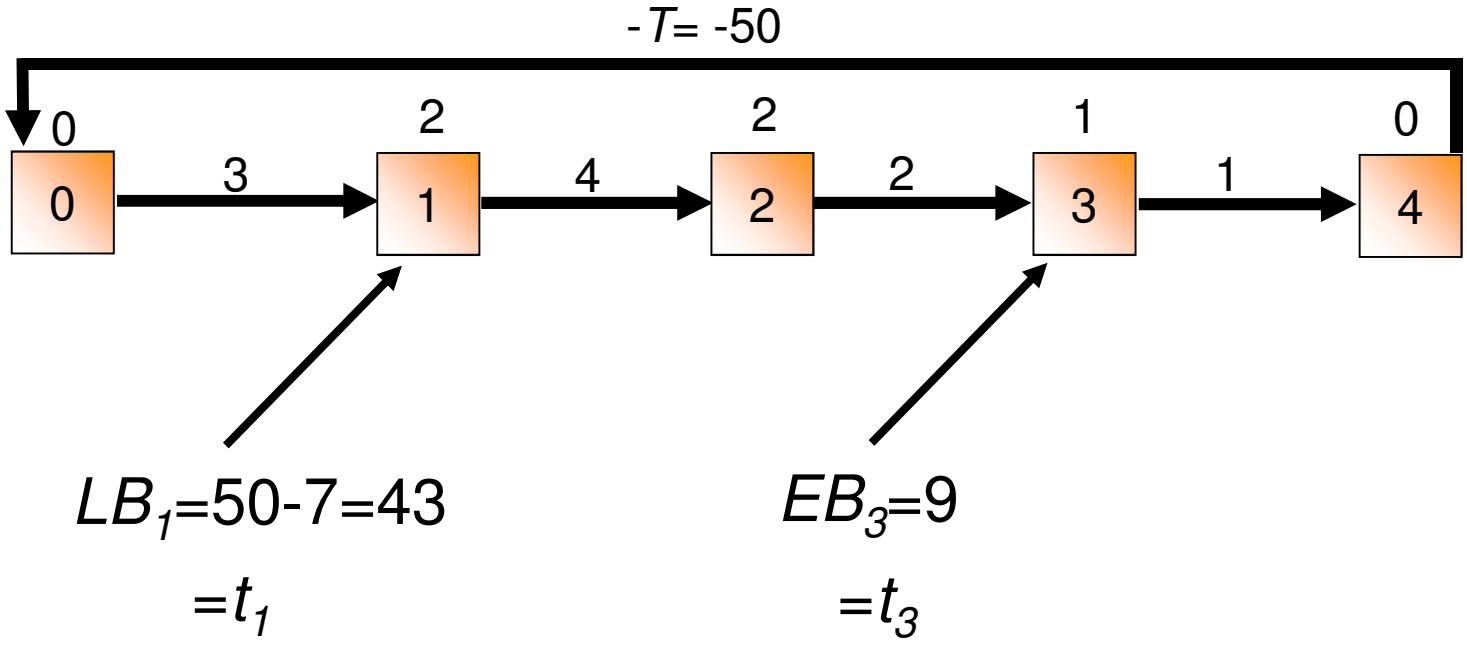
with :

$$t_i^l = \max\left\{LB_h + c_{h,i} | h \in \Gamma^{-1}(i)\right\} \text{ and}$$

$$t_i^u = \min\left\{EB_j - c_{i,j} | j \in \Gamma(i)\right\}$$

- If this interval is large, task *i* can be seen as an uncritical process in the project

Example



BUT: A feasible time table has to fulfill:

$$t_1 + 6 \leq t_3$$

Therefore, it is not feasible!

Summary (buffer times)

Characterization	Buffer time	Identifier	Comment
General time-table-oriented buffer times	TT depending forward buffer	$TFB_i(t)$	Always valid
	TT depending backward buffer	$TBB_i(t)$	Always valid
	Sum of both	$TB_i(t)$	Always valid
Extreme buffer times	Minimum buffer	$MinB_i$	Always valid
	Maximum buffer	$MaxB_i$	Always valid
Specific time-table-oriented buffer times	Total buffer	TBT_i	Only valid if the network contains no cycle
	Free buffer	$FBT_i = TFB_i(t_{EB})$	Always valid
	Free backward buffer	$FBBT_i = TBB_i(t_{LB})$	Always valid
	Independent buffer	IBT_i	Frequently not valid

References for section 2

- Brucker, P.; Knust, S.: Complex Scheduling. 2nd edition, Springer, Berlin, Heidelberg, New York, 2012. (**ISBN-10:** 3-6422-3928-5)
- Neumann, K; et all.: Project Scheduling with Time Windows and Scarce Resources. 2nd Edition. Springer, Berlin, Heidelberg, New York, 2007. (**ISBN-10:** 3-5404-0125-3)
- Ziegler, H.: Minimal and maximal floats in project networks. In Grubström, R.W.; Hinterhuber, H.H. (eds.): Production Economics – Trends and Issues, Amsterdam, pp. 91-97, 1985.
- Ziegler, H.: Projektablaufplanung bei komplexer Ablaufstruktur. PhD-thesis at the University of Paderborn, Paderborn, 1986.