Project management

G. Ferrari Trecate

Dipartimento di Ingegneria Industriale e dell'Informazione
Università degli Studi di Pavia

Industrial Automation

Introduction

Problem : how to schedule activities composing a project such as to avoid useless delays

Def. A project is a set of activities A_i, $i = 1, \dots, m$ with duration $d_i \geq 0$ and a set of precedence relationships

$A_i < A_j \iff A_j$ can start only after A_i has been completed

For each A_i one defines

P_i = set of activities that are immediate predecessors of A_i

Example : organization of a scientific congress

Activity	Description and duration	Immediate predecessors
A_1	Choice of the location and invitation to participants (25 days)	/
A_2	List of accepted invitations (10 days)	A_1
A_3	Reservation of rooms for the sessions (5 days)	A_2
A_4	Hotel reservations (5 days)	A_2
A_5	Organization of the sessions (10 days)	A_3
A_6	Printing of the congress program (10 days)	A_5
A_7	Collection of the congress fees (20 days)	A_2

Activity-on-Arcs (AOA) models

Directed network $G = (V, E, d)$

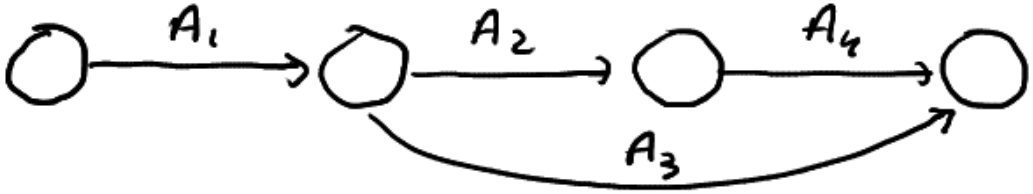
- vertices = start/end instants of activities
- edges = activities
- $A_i < A_j \Leftrightarrow$ there is a path in G passing through A_i and then A_j
- only one start vertex (without predecessors) and one end vertex (without successors). All other nodes have at least a predecessor and at least a successor
- at most one edge between two vertices
- $d(i,j) \geq 0$: labels on edges corresponding to activity durations

Example

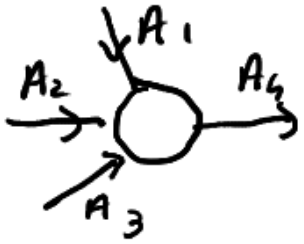
Activities: A_1, A_2, A_3, A_4

Direct precedence relations: $A_1 < A_2$, $A_1 < A_3$, $A_2 < A_4$

↳ AOA network



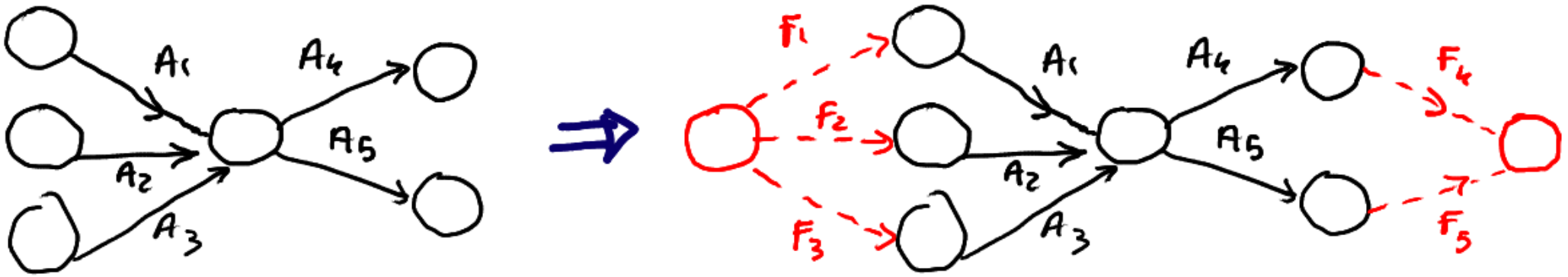
Rmk.



All activities A_1, A_2 and A_3 must be completed before A_4 starts

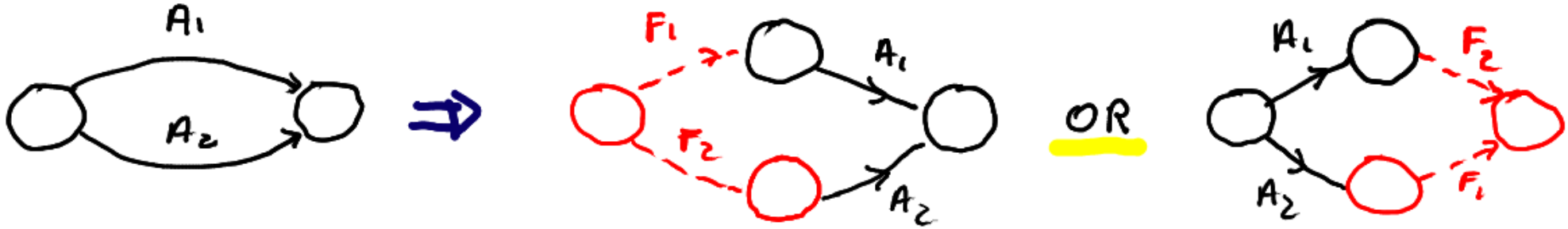
Sometimes it is necessary to add dummy vertices and activities (i.e. with zero duration) to get an AOA network

Case 1: more than a single start/end instant



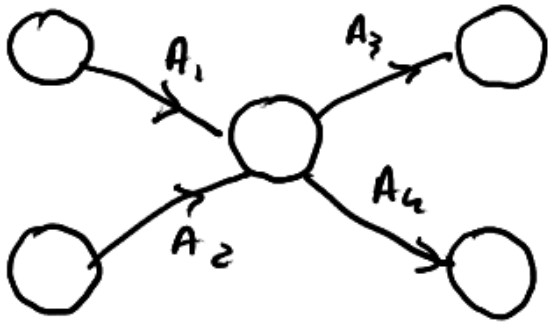
Rmk.
- Dummy vertices and edges do not introduce spurious precedence relations
- Zero-duration activities F_i do not introduce artificial delays in the project completion

Case 2: parallel edges (same start/end instants)

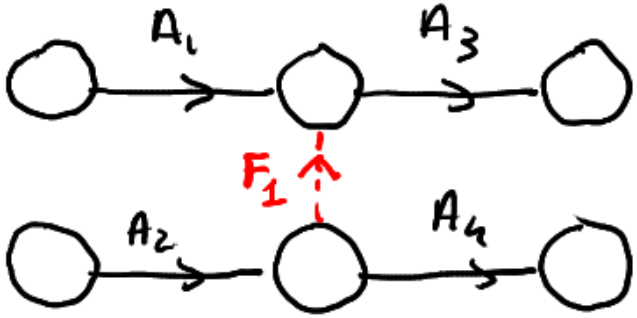


Case 3: precedence constraints involving multiple activities

Ex: $A_1 < A_3$, $A_2 < A_3$, $A_2 \leq A_4$



WRONG: it implies $A_1 < A_4$



OK!

General algorithm for obtaining an AOA network

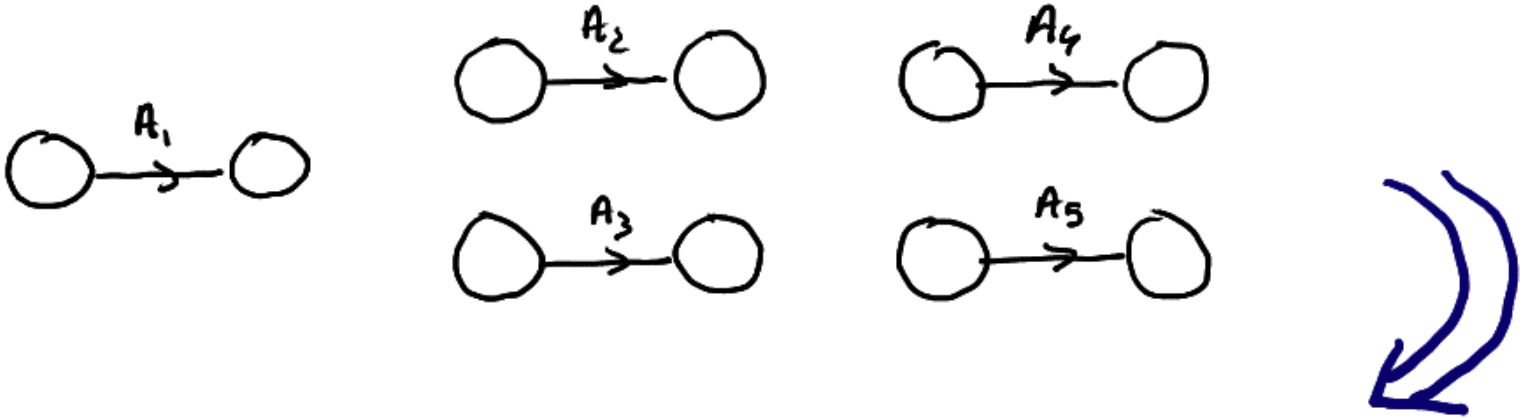
1) Introduce a disjoint edge for each activity
2) Introduce dummy nodes/edges for
   - modeling precedence relations
   - adding start/end vertices, if necessary
3) "Contraction" of dummy activities without creating spurious precedence relations

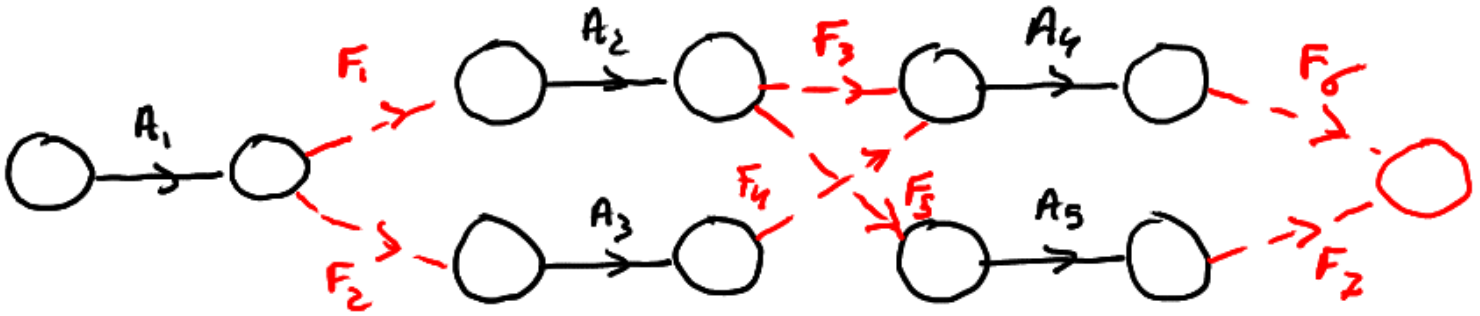
optional step

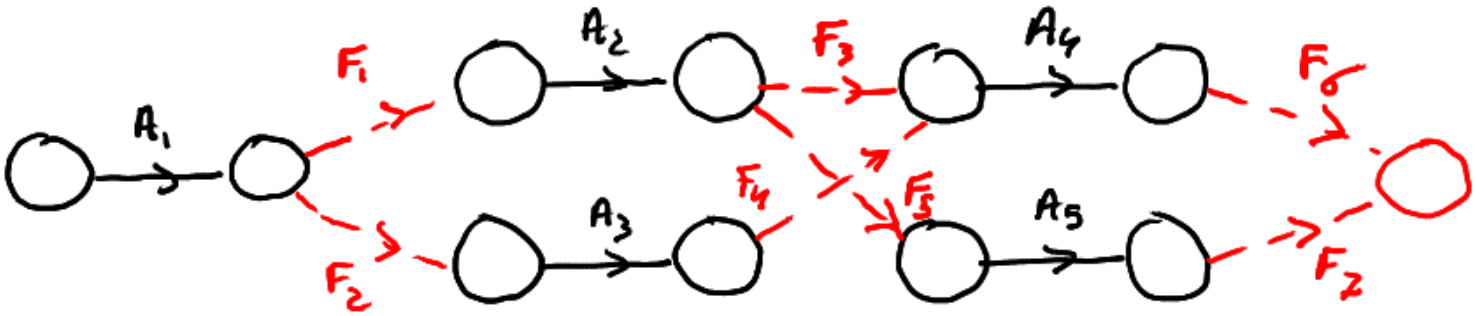
Example

Activities : $A_1, \ldots, A_5$

Precedences : $A_1 < A_2$, $A_1 < A_3$, $A_2 < A_4$, $A_3 < A_4$, $A_2 < A_5$

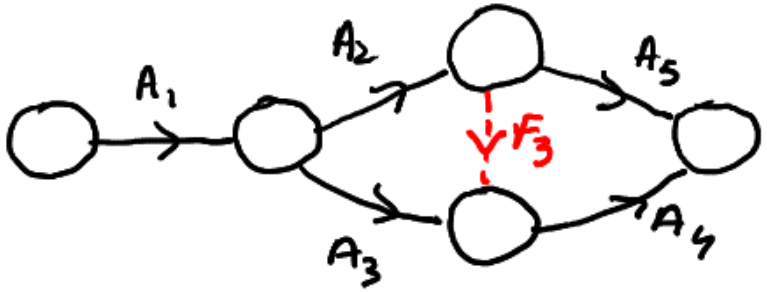




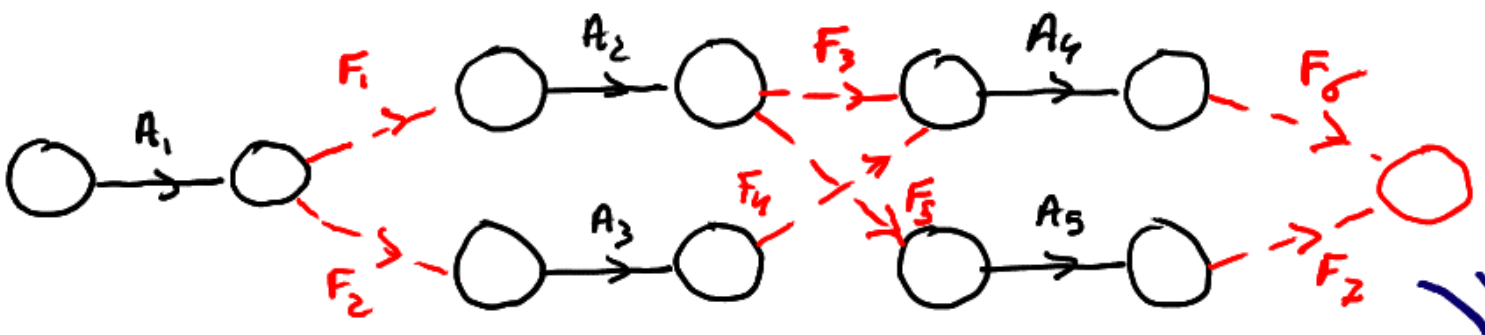


Allowed contractions

$F_1, F_2, F_4, F_5, F_6, F_7$

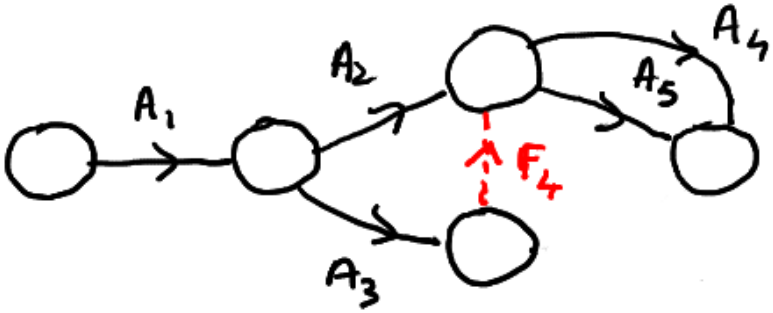


What happens if we contract F_3 instead of F_4?



Allowed contractions

$F_1, F_2, \mathbf{F_3}, F_5, F_6, F_7$



Mistakes

- Spurious precedence $A_3 < A_5$
- Parallel arcs A_4 and A_5

Scheduling problem

Compute feasible start/end times for each activity in order to complete the project in the shortest time.

Key remark. AoA is acyclic by construction.

↳ Critical Path Method (CPM)

CPM algorithm

Input: AOA graph $G = (V, E, d)$ d = activity durations

Step 1. Enumerate vertices such that $(i,j) \in E \Rightarrow i < j$

↳ always possible if G is acyclic

Step 2. For all $v \in V$ let

- E_v be the early event time (the earliest time at which the event v can occour)
- L_v be the late event time (the latest time at which the event v can occour)
- $\mathcal{P}(i)$ be the set of immediate predecessors of node i
- $\mathcal{S}(i)$ " " " " " " successors " " "

Then,

$$E_i = \begin{cases} \max\limits_{v \in \mathcal{P}(i)} E_v + d(v,i) & \text{if } \mathcal{P}(i) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$L_i = \begin{cases} \min\limits_{v \in S(i)} L_v - d(i,v) & \text{if } S(i) \neq \emptyset \\ E_i & \text{otherwise} \end{cases}$$

Rmk.

- Forward computation of E_i from the start vertex
- Backward computation of L_i from the end vertex
- E_i and L_i can be computed in $O(|A|)$, $|A|$ = n° of activities
- By construction, for the end vertex $\bar{v}$, one has $E_{\bar{v}} = L_{\bar{v}}$

Step 3. For each activity (i,j) compute the

- Early Start Time $\quad EST(i,j) = E_i$
- Late Start Time $\quad LST(i,j) = L_j - d(i,j)$

Step 4. An activity (i,j) is critical if $EST(i,j) = LST(i,j)$.
A path from the start vertex to the end vertex is critical if it is composed by critical activities only.
↳ Compute all critical activities and a critical path.

Lemma. There is always a critical path. Moreover a critical path is always a path of maximal duration from the start to the end node

Remarks

- The early event time of the end node is the minimal time required for completing the project

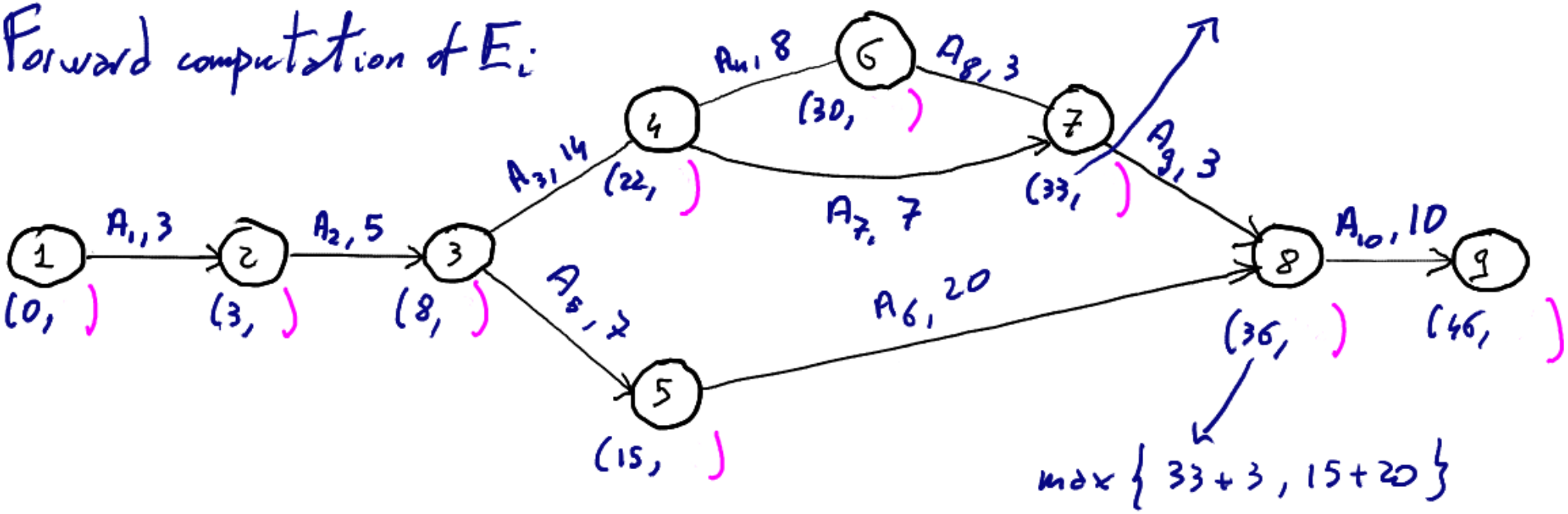
- Any delay in the execution of a critical activity will delay the completion of the project

  ↳ critical activities are bottlenecks that need to be closely monitored

Example

Notation on nodes : (E_i, L_i)

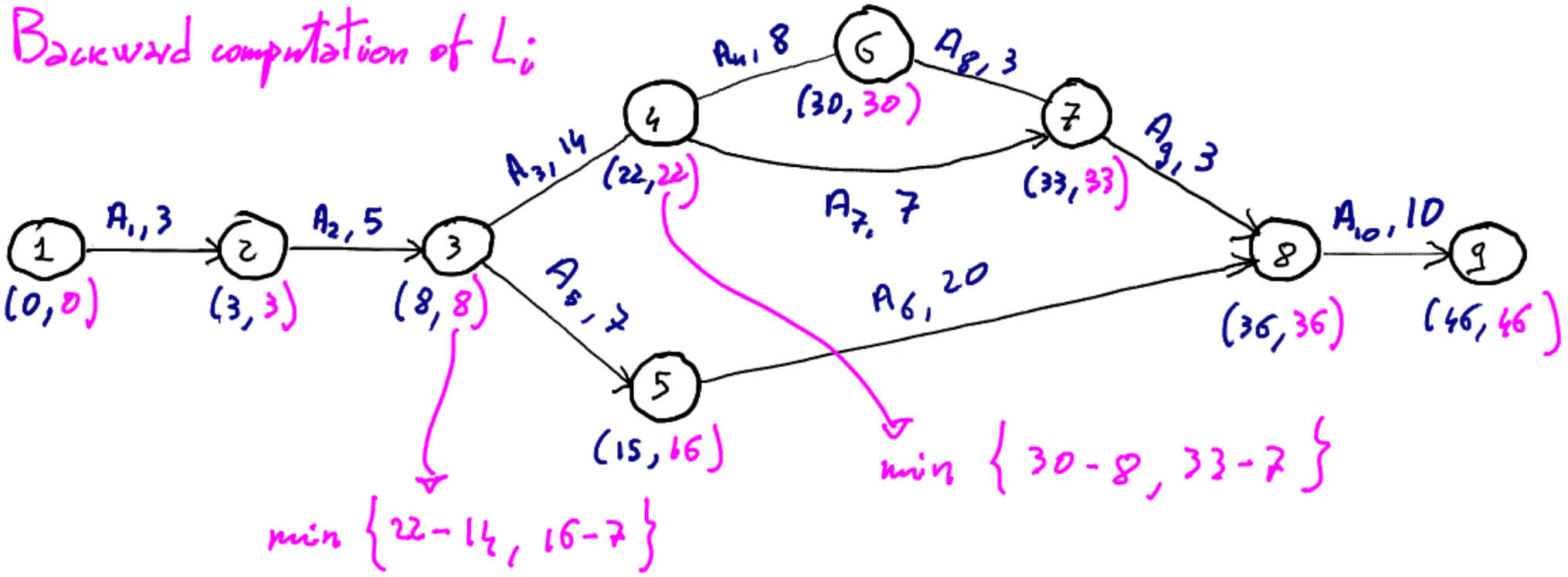
Forward computation of E_i

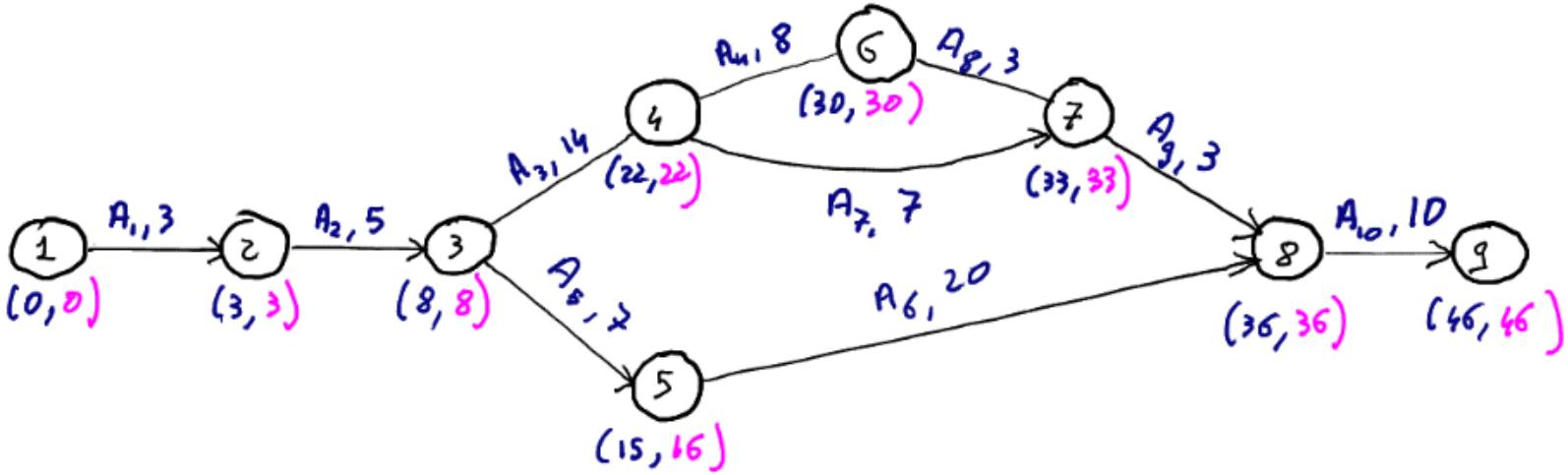


Example

Notation on nodes : (E_i, L_i)

Backward computation of L_i

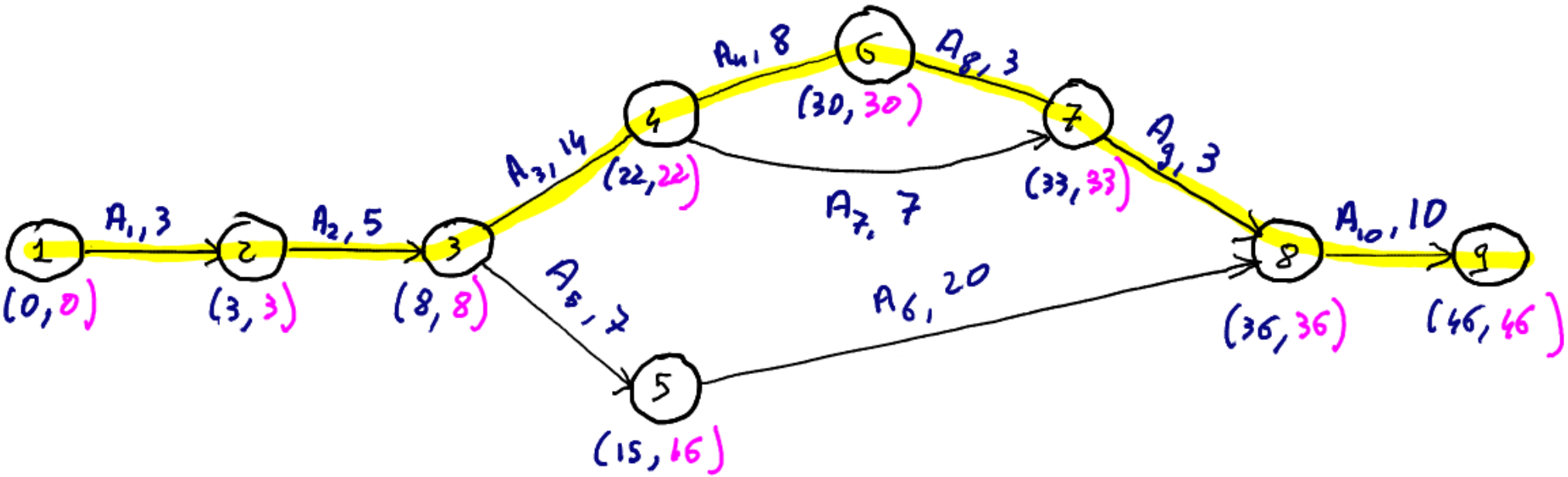




Activity (i,j)	EST E_i	LST $L_j - d(i,j)$	Critical?
A_1	0	0	y
A_2	3	3	y
A_3	8	8	y
A_4	22	22	y
A_5	8	9	N

Activity (i,j)	EST E_i	LST $L_j - d(i,j)$	Critical?
A_6	15	16	N
A_7	22	26	N
A_8	30	30	y
A_9	33	33	y
A_{10}	36	36	y

Critical path: 1, 2, 3, 4, 6, 7, 8, 9



Minimal time for completing the project: 46

Concluding remarks

- Computing a critical path corresponds to computing a shortest path from the start vertex to the end vertex in a network with weights $-d(i,s)$

  ↳ No cycle with negative cost ⇒ Floyd-Warshall can be applied
  ↳ complexity $O(|V|^3)$

- CPM has lower complexity because it exploits the fact that an AOA network is acyclic.