Planning and Scheduling in Maintenance

Cees Witteveen

TU Delft

Bob Huisman

NedTrain

Main topics

Given a set of jobs subject to **time** and **resource** constraints that have to be completed **as soon as possible**

how to solve a scheduling problem

how to decide when to start/complete each job such that
- all constraints (resource + time constraints) are satisfied,
- all objectives are met.

how to distribute a scheduling problem

how to ensure that a scheduling problem can be decomposed and parts can be solved independently.

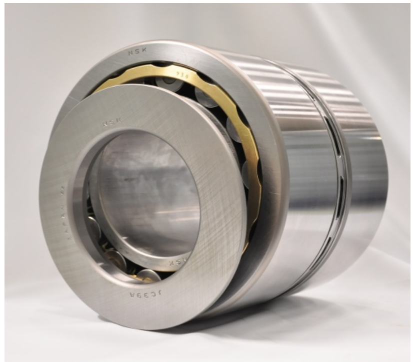
Our global plan

- maintenance scheduling: a simple **example**.
- **specifying** maintenance scheduling problems.
- **solving** scheduling problems.
- **distributing** problems over independent teams.
- **demo** of a tool for maintenance scheduling.

me

Bob

Example: train maintenance



axle integrity inspection

15 min preparation

40 min specialized work

brake repair

25 min preparation

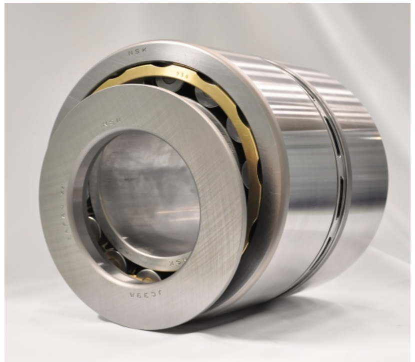
75 min specialized work

wheel profiling

35 min preparation

90 min specialized work

Example: train maintenance



axle integrity inspection

15 min preparation

40 min specialized work

brake repair

25 min preparation

75 min specialized work

wheel profiling

35 min preparation

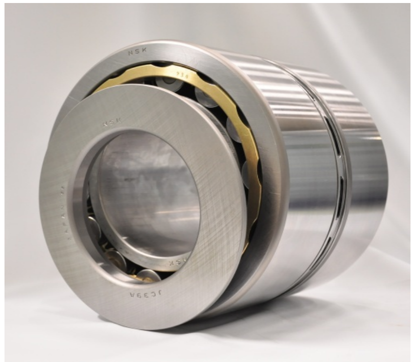
90 min specialized work

jobs

time constraints

resource constraints

Example: train maintenance



axle integrity ir

15 min prep

40 min spec

brake repair

25 min preparation

75 min spec

wheel profiling

35 min prep

90 min spec

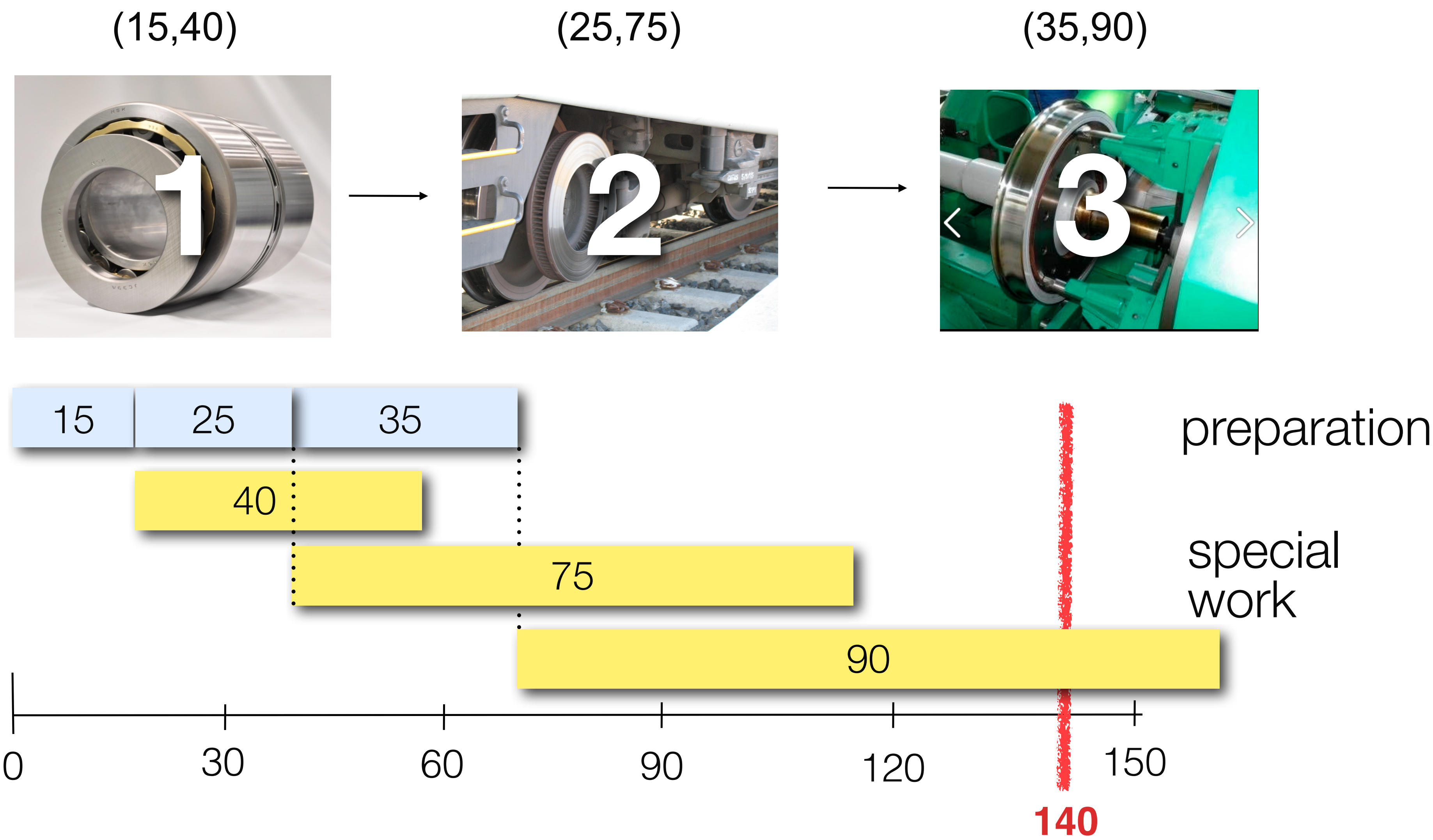
You are the planner ….

1. there is **one** person for doing the preparations.
2. there are plenty of specialists available.

You are required ….

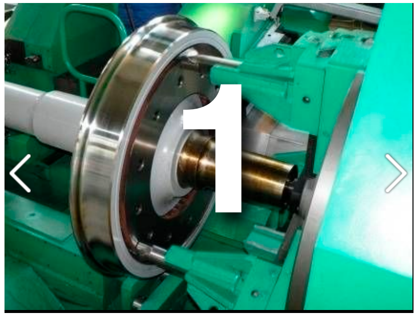
1. to decide **what** to do **when**.
2. to do it **quickly**: you have at most **140** minutes.

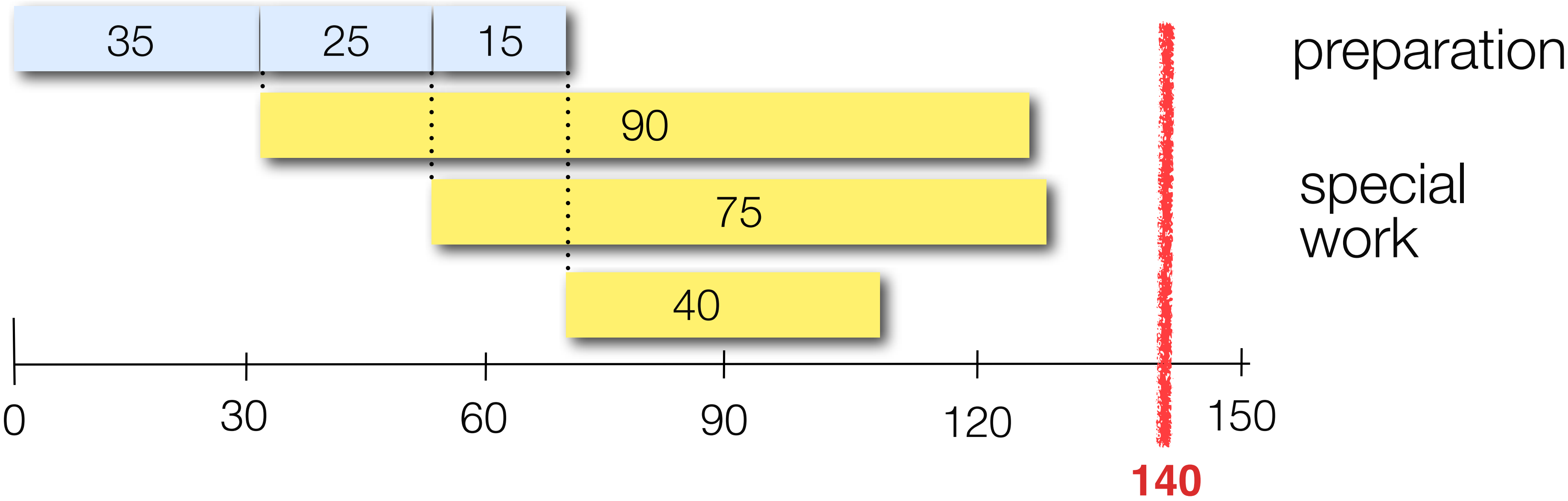
First attempt



Second attempt

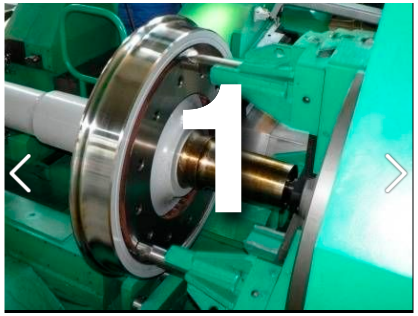
(35,90) (25,75) (15,40)





Scheduling Algorithm

(35,90) (25,75) (15,40)



→

→

scheduling algorithm

begin
  while there is still some job left
    choose the job with longest special work time;
    mark the job as done;
end

this algorithm ensures a **minimal makespan**

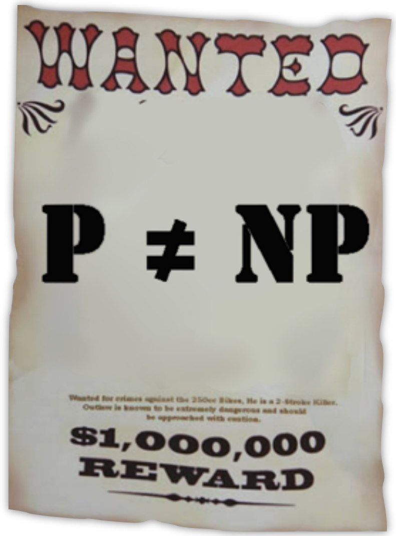
(prove it yourself)

Remarks about the example

- The algorithm we showed is very efficient, but requires an **unbounded number** of specialists.
- If there is only **one** (1) specialist as resource, there is also an efficient algorithm to find the best schedule.
- If there is a bounded number of specialists (>1), we don't know an efficient algorithm, yet.

*In fact, you will earn **$1.000.000** by finding it!*

http://www.claymath.org/millennium/



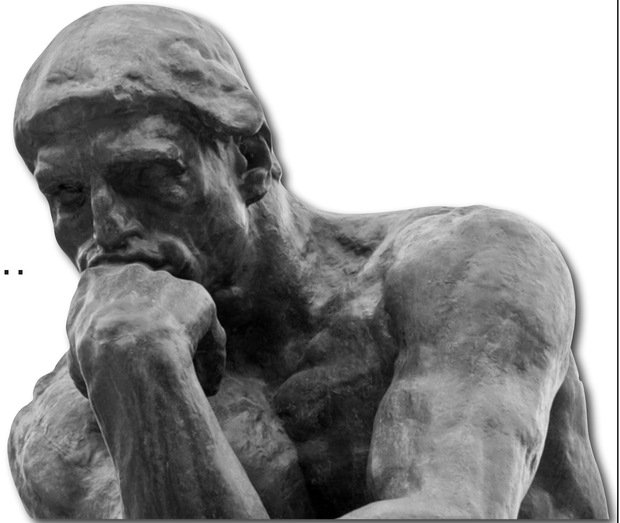
Remarks about the example

- To solve the problem with a bounded number of specialists, we have to rely on **enumerating** all possibilities.
  (10 jobs gives $3{,}6 \times 10^6$ possibilities
  20 jobs gives $2{,}4 \times 10^{18}$ possibilities).

- Such problems are assumed to be intractable problems.

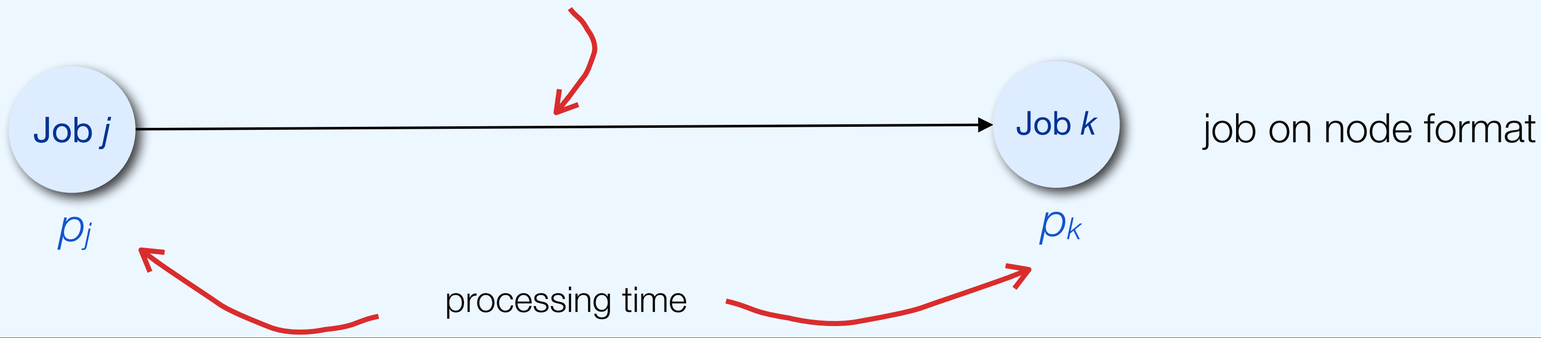
- Most scheduling problems are suspected to be intractable.

That's why we need to think hard to "solve" them ...



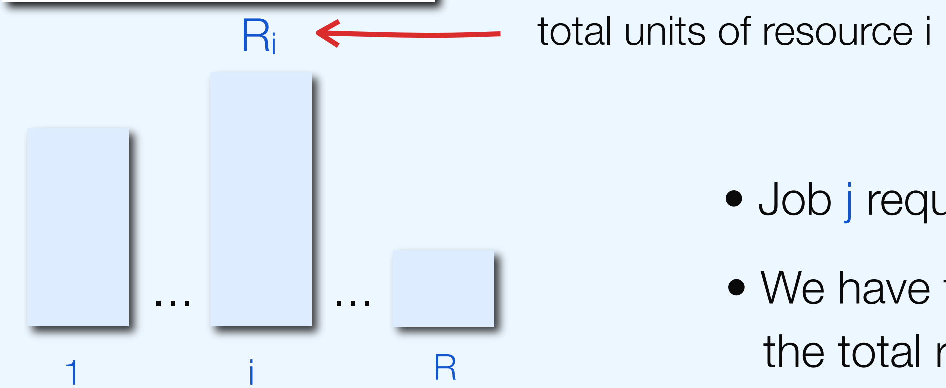
Specificying Scheduling Problems

Time constraints



Resource Constrained Project Scheduling Problem (RCPSP)

Resource constraints

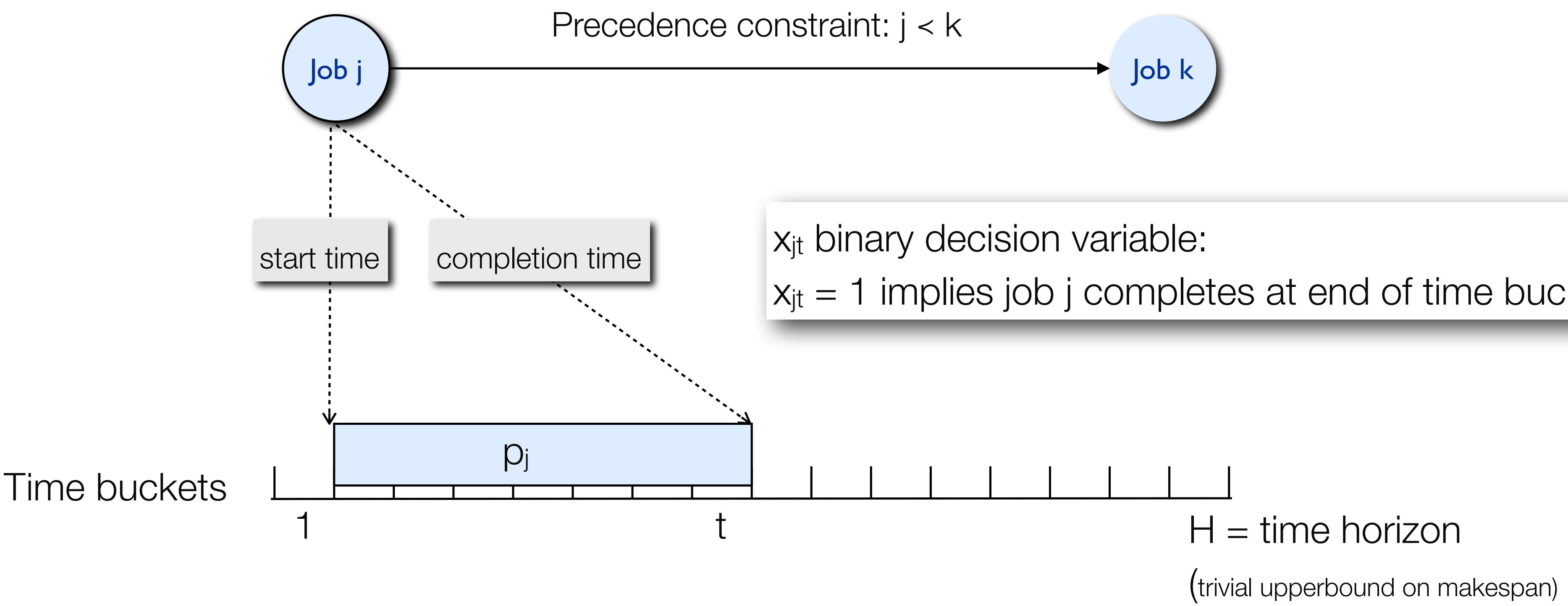


R renewable resource types

- Job j requires R_{ij} units of resource i
- We have to ensure that at any moment of time the total resource consumption does not exceed capacity

TUDelft

RCPSP: Specification (i)



x_{jt} binary decision variable:
$x_{jt} = 1$ implies job j completes at end of time bucket t

- every job j has a unique completion time: $\sum_{t=1}^{H} x_{jt} = 1$
- j < k implies $\sum_{t=1}^{H} t.x_{jt} \; + \; \leq \sum_{t=1}^{H} t.x_{kt} - p_k$
- during any time bucket t, job j uses $R_{ij}.\sum_{u=t}^{t+p_j-1} x_{ju}$ units of resource i

Solving the RCPSP as an IP

minimize $\displaystyle\sum_{t=1}^{H} t \cdot x_{n+1,t}$

subject to $\displaystyle\sum_{t=1}^{H} t \cdot x_{jt} + p_k - \sum_{t=1}^{H} t x_{kt} \leq 0$ for all j, k s.t. $j < k$

$$\sum_{j=1}^{n}\left(R_{ij}\sum_{u=t}^{t+p_j-1} x_{ju}\right) \leq R_i \qquad i = 1,\dots R;\ t = 1,\dots,H$$

$$\sum_{t=1}^{H} x_{jt} = 1$$

- $x_{i,t}$ is a binary variable taking value *1* if job *j* is completed at time *t* and 0 else,
- $x_{n+1,t}$ is a dummy job such that all jobs precede it,
- H is an upperbound on the makespan.

Scheduling: complexity

Problem specification 1

NP-hard problem

- set of linear time constraints
- set of resource constraints
- objective: minimize makespan

Resource Constrained Project Scheduling Problem

(RCPSP)

use heuristics to translate resource constraints into time constraints

Problem specification 2

Efficiently solvable problem

- set of linear time constraints
- ~~set of resource constraints~~
- objective: minimize makespan

Simple Temporal Problem

(STP)

Our approach to solve RCPSP

Next, Bob discusses the reduction process

I concentrate on this problem first

resource constraints
+
time constraints

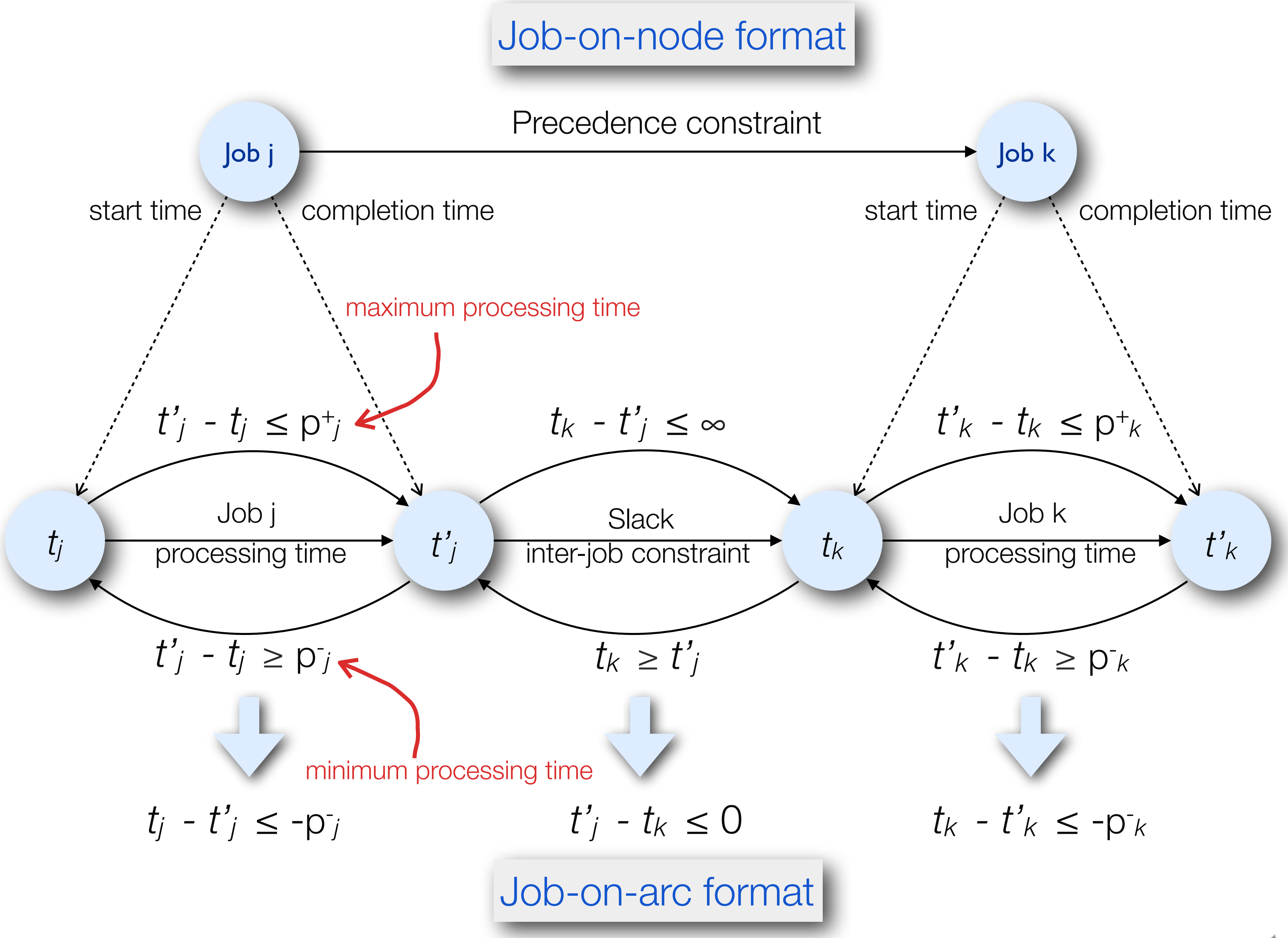
translate resource constraints into time constraints

time constraints

Simple Temporal Problem

(STP)

From jobs to time points



Simple Temporal Problems

A Simple Temporal Problem (**STP**) is a tuple $S = (T, C)$ where

- T is a set of (time) variables
- C is a set of binary constraints of the form $t' - t \leq \delta$, where δ is an arbitrary constant.

All Constraints are linear !

A *solution* to an **STP** $S = (T, C)$ is an *assignment* $\{ t_i = v_i \; : t_i \in T \}$ of values (times) to all variables in T such that all constraints in C are satisfied.

Small example

Railway maintenance requires two jobs A and B to schedule. Job A arrives at 12.00 hrs and requires at most 5 minutes. Job B should start at least 2 minutes later than A, and must be completed within 3 minutes after completion of A. B requires at most 7 minutes and at least 2 minutes to process.

When to start A, and when to start B?

variables

t_A, t_B : start of A, B

t'_A, t'_B : completion of A, B

in our case z_0 refers to 12.00 hrs; we take this as our reference point **0**

constraints (w.r.t.time reference z_0)

$z_0 = 0$	$t'_A - t_A \leq 5$	$t'_B - t'_A \leq 3$
$t_A - z_0 \leq 0$	$t_A - t'_A \leq 0$	$t'_B - t_B \leq 7$
$z_0 - t_A \leq 0$	$t_A - t_B \leq -2$	$t_B - t'_B \leq -2$

Small example: STP

Railway maintenance requires two jobs A and B to schedule.
Job A arrives at 12.00 hrs and requires at most 5 minutes.
Job B should start at least 2 minutes later than A, and must be completed within 3 minutes after completion of A.
B requires at most 7 minutes and at least 2 minutes to process.

When to start A, and when to start B?

variables

t_A, t_B : start of A, B

t'_A, t'_B : completion of A, B

constraints (w.r.t.time reference z_0)

$z_0 = 0$	$t'_A - t_A \leq 5$	$t'_B - t'_A \leq 3$
$t_A - z_0 \leq 0$	$t_A - t'_A \leq 0$	$t'_B - t_B \leq 7$
$z_0 - t_A \leq 0$	$t_A - t_B \leq -2$	$t_B - t'_B \leq -2$

This is the standard representation in
Simple
Temporal
Problem

TU Delft

Small example: Solution?

Railway maintenance requires two jobs A and B to schedule.
Job A arrives at 12.00 hrs and requires at most 5 minutes.
Job B should start at least 2 minutes later than A, and
must be complet…
B requires at m…
to process.

When to start …

- How to find solutions in genera…

By trial and error:

variables	
t_A, t_B	: start of A, B
t'_A, t'_B	: completion of A, B

constraints (w.r.t.time reference z_0)

$z_0 = 0$	$t'_A - t_A \leq 5$	$t'_B - t'_A \leq 3$
$t_A - z_0 \leq 0$	$t_A - t'_A \leq 0$	$t'_B - t_B \leq 7$
$z_0 - t_A \leq 0$	$t_A - t_B \leq -2$	$t_B - t'_B \leq -2$

	Solution 1:	Solution 2:
z_0	0	0
t_A	0	0
t'_A	1	5
t_B	2	6
t'_B	4	8
	A starts at 12:00, B starts at 12:02	A starts at 12:00, B starts at 12:06

Small example: STN

Railway maintenance requires two jobs A and B to schedule.
Job A arrives at 12.00 hrs and requires at most 5 minutes.
Job B should start at least 2 minutes later than A, and must be completed within 3 minutes after completion of A.
B requires at most 7 minutes and at least 2 minutes to process.

When to start A, and when to start B?

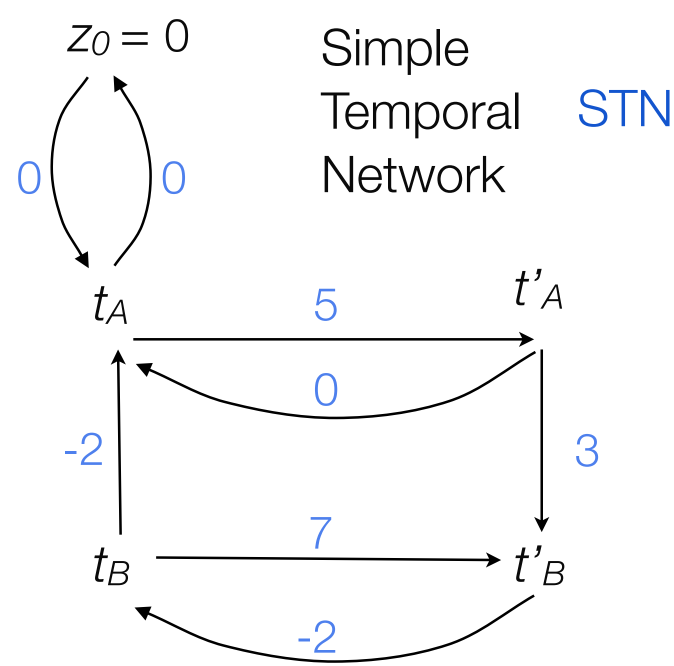
variables

t_A, t_B : start of A, B

t'_A, t'_B : completion of A, B

constraints (w.r.t.time reference z_0)

$z_0 = 0$	$t'_A - t_A \leq 5$	$t'_B - t'_A \leq 3$
$t_A - z_0 \leq 0$	$t_A - t'_A \leq 0$	$t'_B - t_B \leq 7$
$z_0 - t_A \leq 0$	$t_A - t_B \leq -2$	$t_B - t'_B \leq -2$



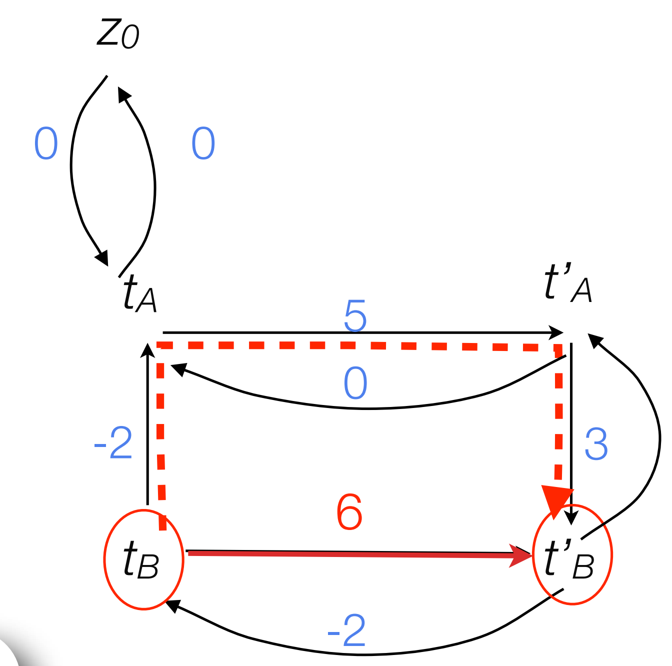
implied constraints

explicit constraints can be combined to imply other constraints:

constraints	some implied constraints
$t'_A - t_A \leq 5$ $t_A - t_B \leq -2$ $t_B - t'_B \leq -2$ $t'_B - t'_A \leq 3$ $t'_B - t_B \leq 7$	$t'_B - t_B = (t'_B - t'_A) + (t'_A - t_A) + (t_A - t_B)$ $\leq 3 + 5 + -2$ $= 6$

Note that the implied constraint is stronger, replacing the original constraint

strongest constraints in STP correspond to **shortest paths** in STN



Look at the labels of the edges as **distances** between the corresponding nodes.

Finding an **alternative path** comes down to identifying **another constraint**!

TUDelft

implied constraints

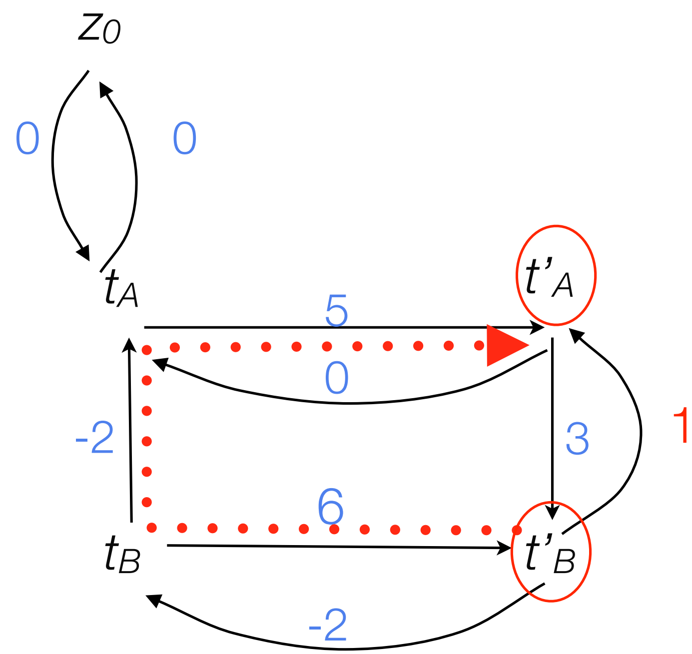
explicit constraints can be combined to imply other constraints:

constraints
$t'_A - t_A \leq 5$
$t_A - t_B \leq -2$
$t_B - t'_B \leq -2$
$t'_B - t'_A \leq 3$
$t'_B - t_B \leq 7$

some implied constraints

$$t'_B - t_B = (t'_B - t'_A) + (t'_A - t_A) + (t_A - t_B)$$
$$\leq 3 + 5 + -2$$
$$= 6$$

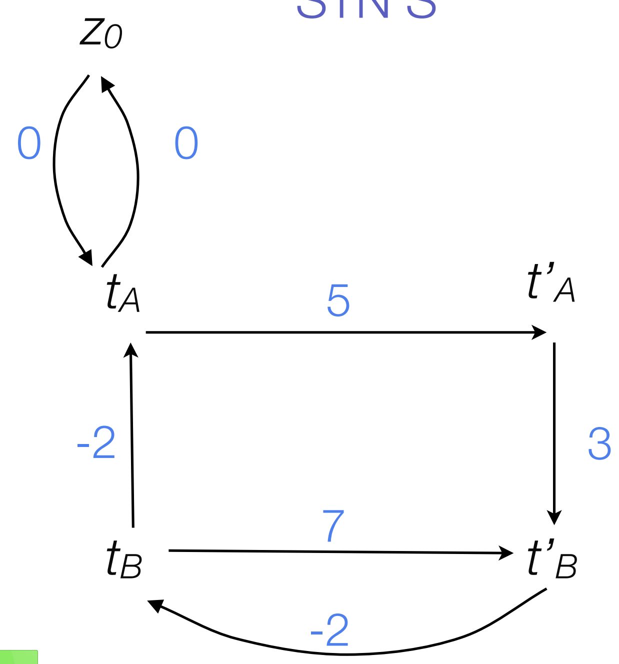
$$t'_A - t'_B = (t'_A - t_A) + (t_A - t_B) + (t_B - t'_B)$$
$$\leq 5 + -2 + -2$$
$$= 1$$



strongest constraints in STP correspond to **shortest paths** in STN

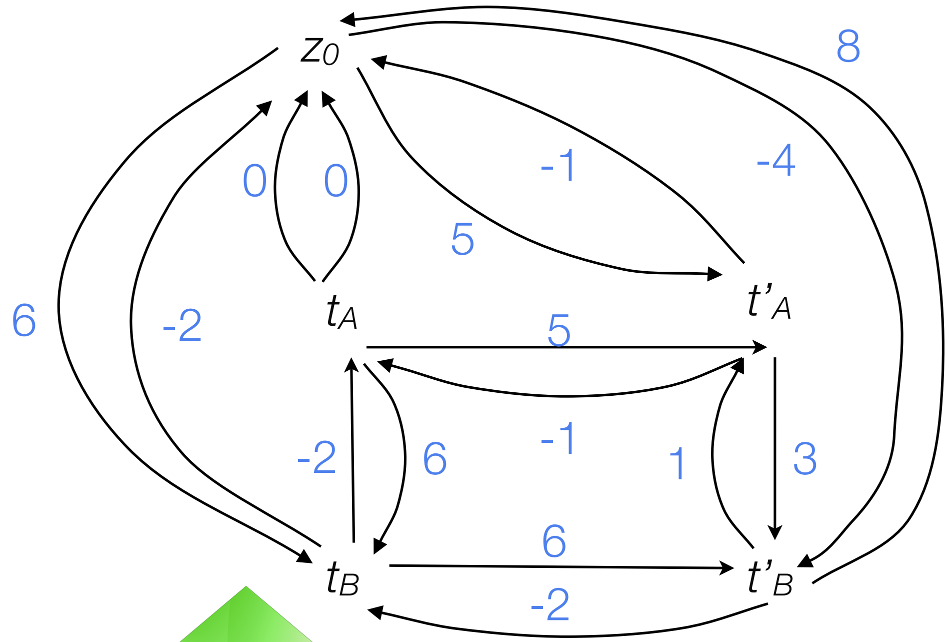
shortest paths can be computed between any pair of nodes in the STN

STN S



equivalent systems

minimal STN S'



constraint matrix

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	∞	∞
t_A	0	0	5	6	8
t'_A	∞	∞	0	∞	3
t_B	∞	-2	∞	0	7
t'_B	∞	∞	∞	-2	0

all-pairs shortest path computation

distance matrix

D:

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

checking consistency

Given an STP S with its graph G_S and distance graph D_S, the following statements are equivalent:

- S is consistent
- D_S has no negative entries on its diagonal
- G_S has no negative cycles

this property can be checked in $O(n^3)$

original constraint matrix

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	∞	∞
t_A	0	0	5	6	8
t'_A	∞	∞	0	∞	3
t_B	∞	-2	∞	0	7
t'_B	∞	∞	∞	-2	0

all-pairs shortest path computation $O(n^3)$

distance matrix $D_S = [\ d(t,t')\ ]$

S is consistent

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

Finding solutions

Given an STN S with its distance graph D_S, If D_S does not contain negative diagonal elements, then

- the set $\{ t_i = D(z_0, t_i) \ : i=1,2, \ldots n\} \cup \{z_0 = 0\}$ is a solution — latest starting times
- the set $\{ t_i = -D(t_i, z_0) : i=1,2, \ldots n\} \cup \{z_0 = 0\}$ is also a solution — earliest starting times

Why ?

original constraint matrix

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	∞	∞
t_A	0	0	5	6	8
t'_A	∞	∞	0	∞	3
t_B	∞	-2	∞	0	7
t'_B	∞	∞	∞	-2	0

all-pairs shortest path computation $O(n^3)$

two solutions

distance matrix $D_S = [d(t,t')]$

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

Finding solutions

Given an STN S with its distance graph D_S, If D_S does not contain negative diagonal elements, then

- the set $\{ t_i = D(z_0, t_i) : i=1,2, \dots n\} \cup \{z_0 = 0\}$ is a solution
- the set $\{ t_i = -D(t_i, z_0) : i=1,2, \dots n\} \cup \{z_0 = 0\}$ is also a solution

Why ?

Take an arbitrary constraint $t' - t \leq \delta$. We have

$$D(z_0,t') \leq D(z_0,t) + D(t,t') \leq D(z_0,t) + \delta$$

Hence, $D(z_0,t') - D(z_0,t) \leq \delta$. So the first set of solutions satisfies every constraint.

Likewise,

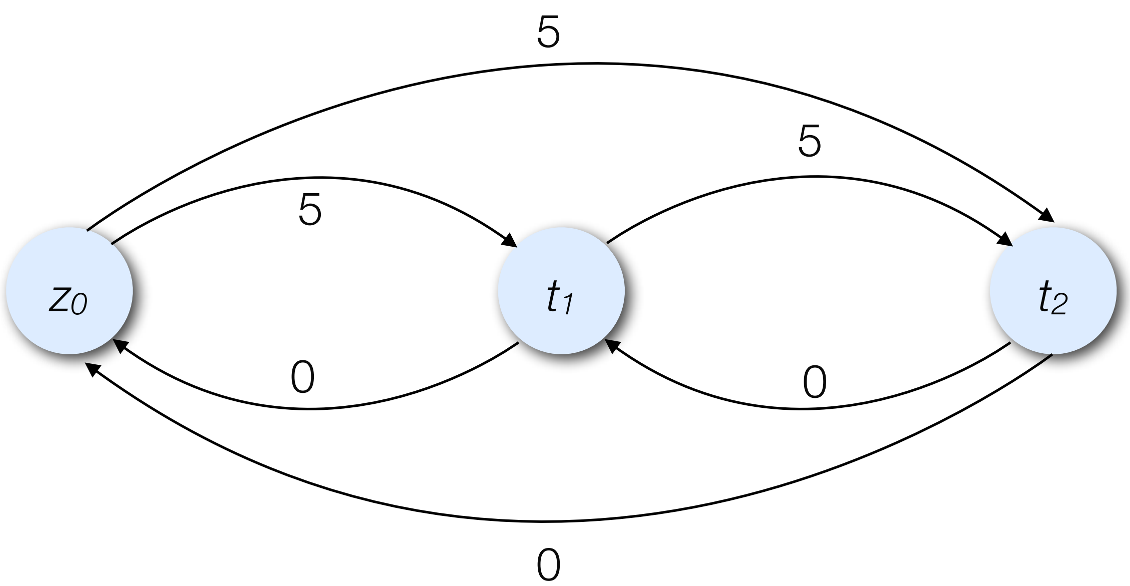
$$D(t,z_0) \leq D(t,t') + D(t',z_0) \leq \delta + D(t',z_0)$$

Hence, $D(t,z_0) - D(t',z_0) \leq \delta$. So the second set of solutions satisfies every constraint, too.

distance matrix $D_S = [d(t,t')]$

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	0	5	6	8
t'_A	-1	-1	0	1	3
t_B	-2	-2	3	0	6
t'_B	-4	-4	1	-2	0

Finding solutions: Example



STN (minimal)
encoding two events $t_1 \leq t_2$
starting in [0,5].

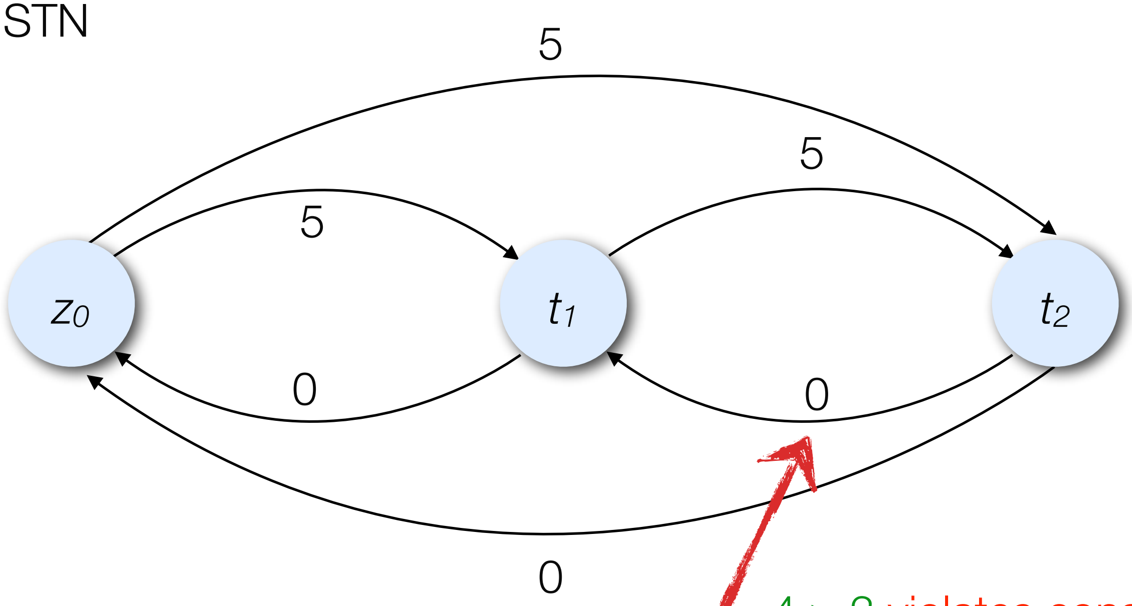
	earliest	latest
z_0	0	0
t_1	0	5
t_2	0	5

for every schedule σ: values σ(t) occur between **earliest** and **latest** starting times of t.

but not all values between earliest and latest starting times constitute a schedule !

Finding solutions: Example

STN



4 > 3 violates constraint $t_1 - t_2 \leq 0$

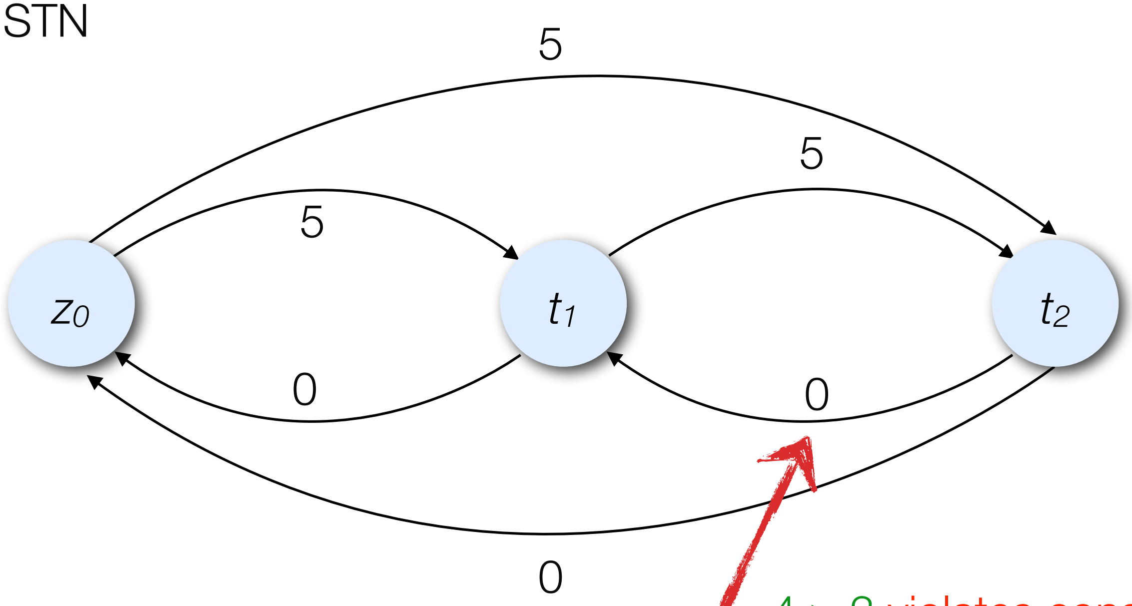
	earliest		latest
z_0	0	0	0
t_1	0	4	5
t_2	0	3	5

for every schedule σ: values σ(t) occur between earliest and latest starting times of t.

but not all values between earliest and latest starting times constitute a schedule !

Finding solutions: Example

STN



Lesson learnt:

after choosing a value v for a variable t, add the constraint t = v to the STN and **recalculate** the minimal STN

4 > 3 violates constraint $t_1 - t_2 \leq 0$

	earliest		latest
z_0	0	0	0
t_1	0	4	5
t_2	0	3	5

for every schedule σ: values σ(t) occur between earliest and latest starting times of t.

but not all values between earliest and latest starting times constitute a schedule !

Finding solutions

sol := ∅
while $T \neq \varnothing$
  select some t and $v \in [-D(t,z_0), D(z_0,t)]$;
  add constraint $\{t = v\}$ to STN and update the distance matrix D_S;
  remove row and column t from D_S ;
  sol := sol ∪ {t=v};

start with $t_A \in [0,0]$; add $t_A = 0$

distance matrix $D_S = [d(t,t')]$

	z_0	t_A	t'_A	t_B	t'_B
z_0	0	0	5	6	8
t_A	0	~~0~~	~~5~~	~~6~~	~~8~~
t'_A	-1	~~-1~~	0	1	3
t_B	-2	~~-2~~	3	0	6
t'_B	-4	~~-4~~	1	-2	0

Finding solutions

sol := ∅
while $T \neq \emptyset$
  select some t and $v \in [-D(t,z_0), D(z_0,t)]$;
  add constraint $\{t = v\}$ to STN and update the distance matrix D_S;
  remove row and column t from D;
  sol := sol ∪ {t=v}

start with $t_A \in [\,0,0\,]$; add $t_A = 0$
take $t_B \in [\,2, 6\,]$; add $t_B = 4$; update

distance matrix $D_S = [d(t,t')]$

	z_0	t_A	t'_A	t_B	t'_B
z_0	0		5	6	8
t_A					
t'_A	-1		0	1	3
t_B	-2		3	0	6
t'_B	-4		1	-2	0

Finding solutions

sol := ∅
while $T \neq \varnothing$
  select some t and $v \in [-D(t,z_0),\ D(z_0,t)]$;
  add constraint $\{t = v\}$ to STN and update the distance matrix D_S;
  remove row and column t from D;
  sol := sol ∪ {t=v}

start with $t_A \in [\,0,0\,]$; add $t_A = 0$

take $t_B \in [\,2, 6\,]$; add $t_B = 4$; update

distance matrix $D_S = [d(t,t')]$

	z_0	t_A	t'_A	t_B	t'_B
z_0	0		5	4	8
t_A					
t'_A	-1		0	1	3
t_B	-4		3	0	6
t'_B	-4		1	-2	0

Finding solutions

sol := ∅
while $T \neq \varnothing$
  select some t and $v \in [-D(t,z_0), D(z_0,t)]$;
  add constraint $\{t = v\}$ to STN and update the distance matrix D_S;
  remove row and column t from D;
  sol := sol ∪ {t=v}

start with $t_A \in [0,0]$; add $t_A = 0$

take $t_B \in [2, 6]$; add $t_B = 4$; update

distance matrix $D_S = [d(t,t')]$

	z_0	t_A	t'_A	t_B	t'_B
z_0	0		5		8
t_A					
t'_A	-3		0		3
t_B					
t'_B	-6		-1		0

Finding solutions

sol := ∅
while $T \neq \varnothing$
  select some t and $v \in [-D(t,z_0), D(z_0,t)]$;
  add constraint $\{t = v\}$ to STN and update the distance matrix D_S;
  remove row and column t from D;
  sol := sol ∪ {t=v}

distance matrix $D_S = [d(t,t')]$

start with $t_A \in [0,0]$; add $t_A = 0$
take $t_B \in [2, 6]$; add $t_B = 4$; update
take $t'_A \in [3, 5]$; add $t'_A = 3$; update

	z_0	t_A	t'_A	t_B	t'_B
z_0	0		5		8
t_A					
t'_A	-3		0		3
t_B					
t'_B	-6		-1		0

Finding solutions

sol := ∅
while $T \neq \emptyset$
  select some t and $v \in [-D(t,z_0),\ D(z_0,t)]$;
  add constraint $\{t = v\}$ to STN and update the distance matrix D_S;
  remove row and column t from D;
  sol := sol ∪ {t=v}

start with $t_A \in [\,0,0\,]$; add $t_A = 0$
take $t_B \in [\,2, 6\,]$; add $t_B = 4$; update
take $t'_A \in [\,3, 5\,]$; add $t'_A = 3$; update

distance matrix $D_S = [d(t,t')]$

	z_0	t_A	t'_A	t_B	t'_B
z_0	0		3		8
t_A					
t'_A	-3		0		3
t_B					
t'_B	-6		-1		0

Finding solutions

```
sol := ∅
while T ≠ ∅
   select some t and  v ∈ [-D(t,z0), D(z0,t)];
   add constraint {t = v} to STN and update the distance matrix DS;
   remove row and column t from D;
   sol := sol ∪ {t=v}
```

start with $t_A \in [0,0]$; add $t_A = 0$

take $t_B \in [2, 6]$; add $t_B = 4$; update

take $t'_A \in [3, 5]$; add $t'_A = 3$; update

take $t'_B \in [6, 6]$; add $t'_B = 6$

solution s

distance matrix $D_S = [d(t,t')]$

	z_0	t_A	t'_A	t_B	t'_B
z_0	0				6
t_A					
t'_A					
t_B					
t'_B	-6				0

TU Delft

What did we achieve

- we introduced a (maintenance) scheduling problem as a constraint problem: set of jobs subject to time + resource constraints that have to be completed as early as possible.
- we showed how to solve the problem if only time constraints are present (STP)
- now **Bob** shows how to take the resource constraints into account…
- ... that will happen after the break.

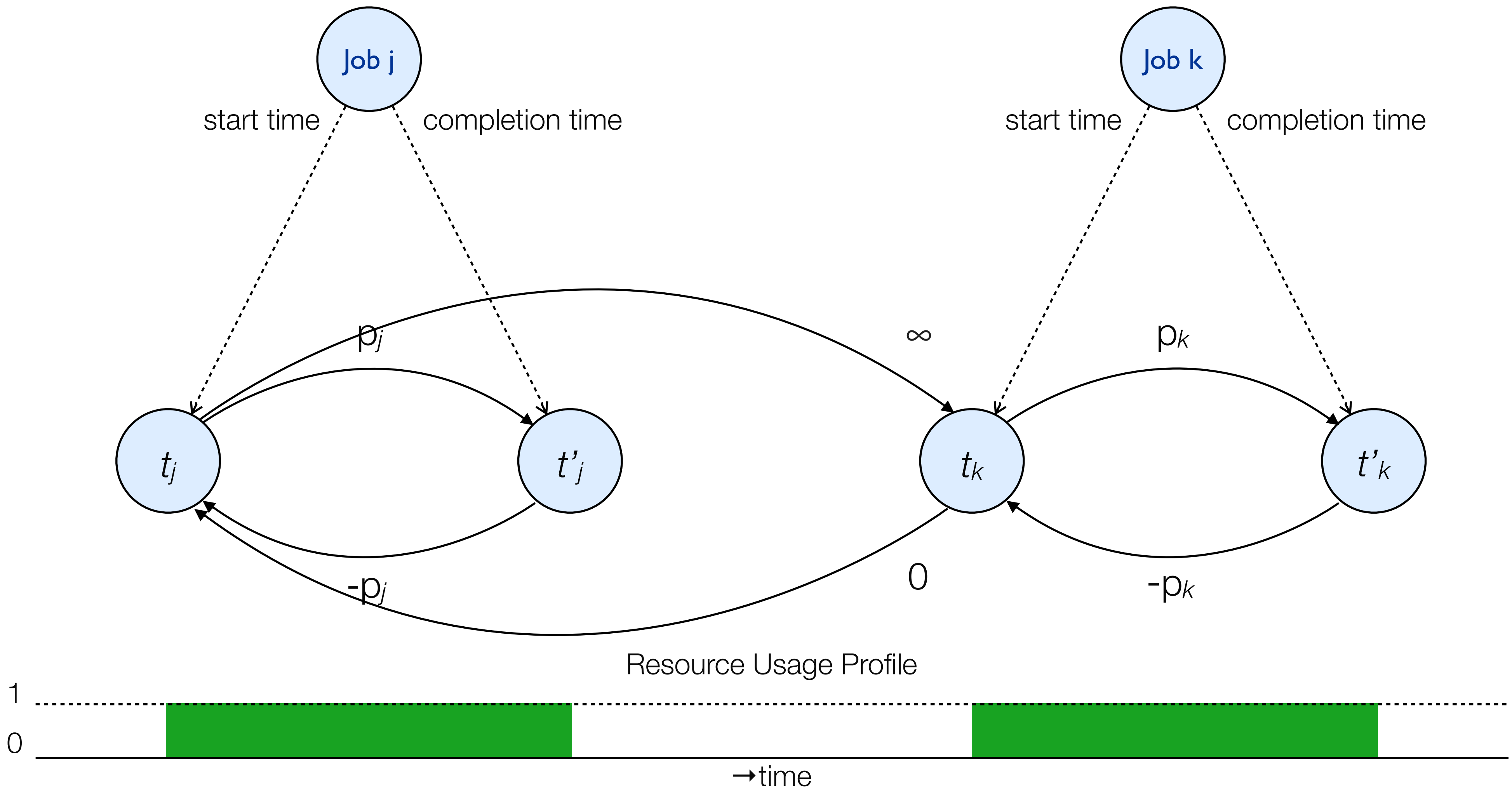
From RCPSP to STP

Finding solutions by constraint posting

Resource usage

Resource availability: $R_1 = 1$

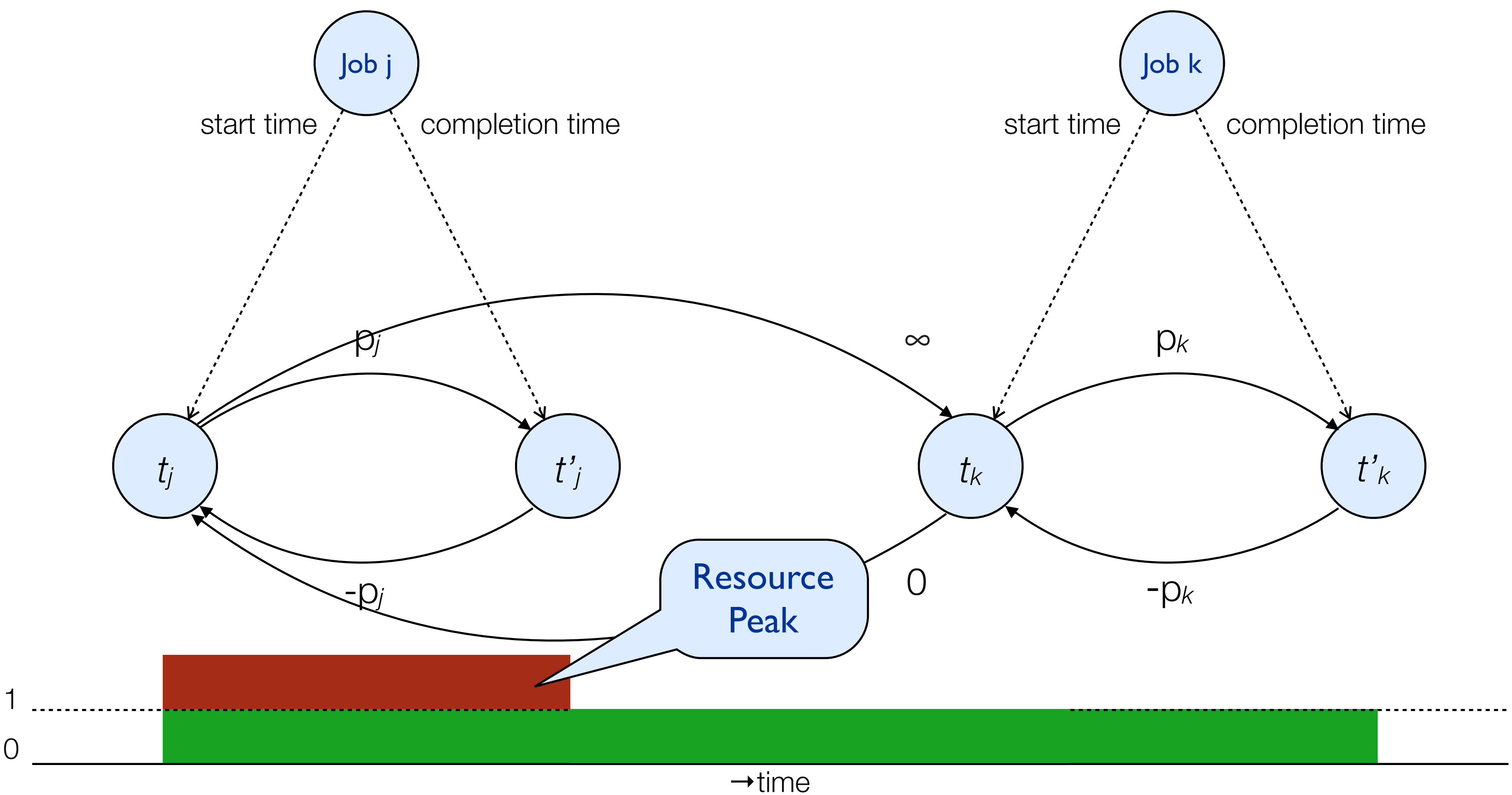
Resource requirements: $R_{1j} = 1$ $R_{1k} = 1$



Resource usage

Resource availability: $R_1 = 1$

Resource requirements: $R_{1j} = 1$ $R_{1k} = 1$

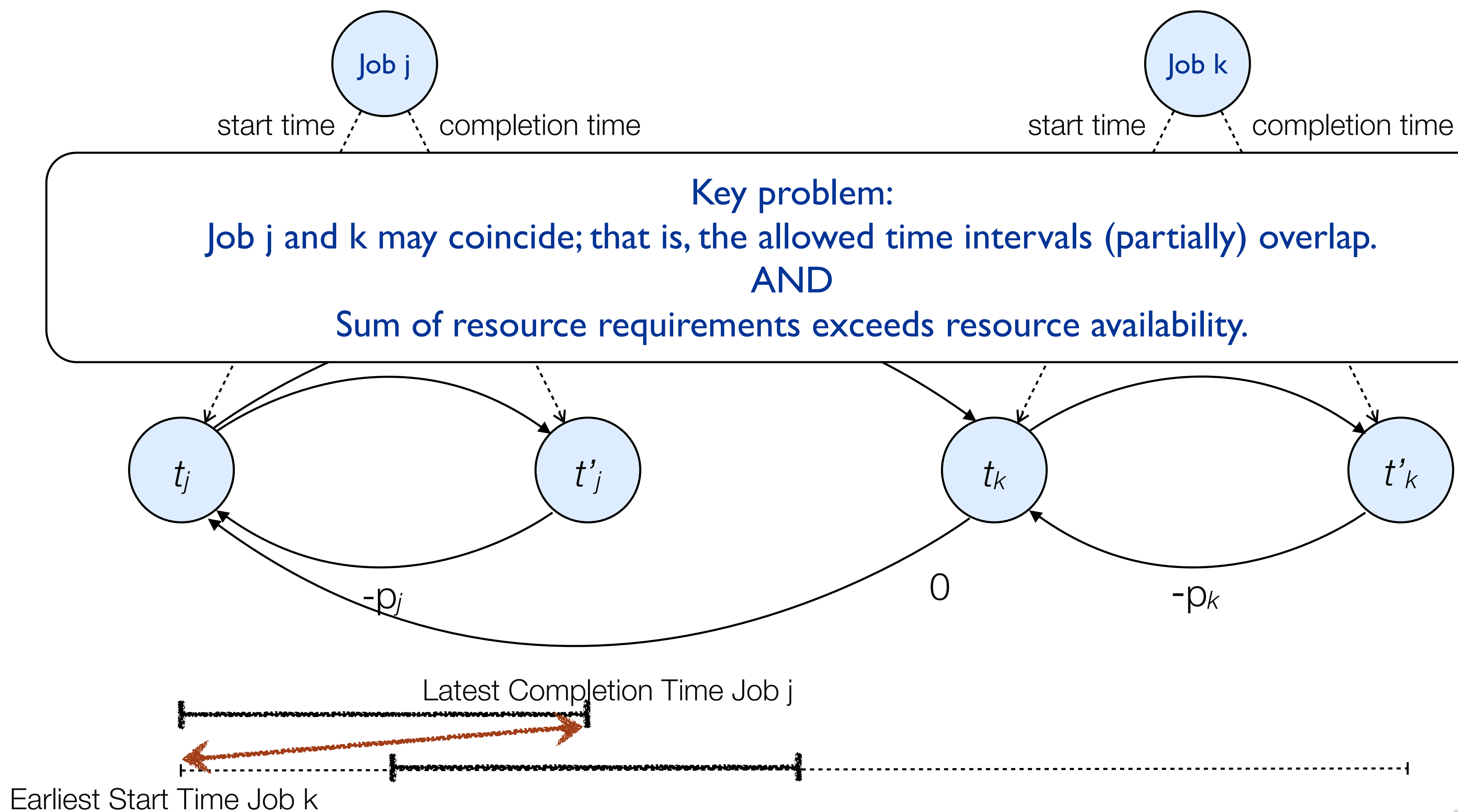


TU Delft

Resource usage

Resource availability: $R_1 = 1$

Resource requirements: $R_{1j} = 1$ $R_{1k} = 1$



Key problem:
Job j and k may coincide; that is, the allowed time intervals (partially) overlap.
AND
Sum of resource requirements exceeds resource availability.

Resource usage

Resource availability: $R_1 = 1$

Resource requirements: $R_{1j} = 1$ $R_{1k} = 1$

Job j

start time completion time

Job k

start time completion time

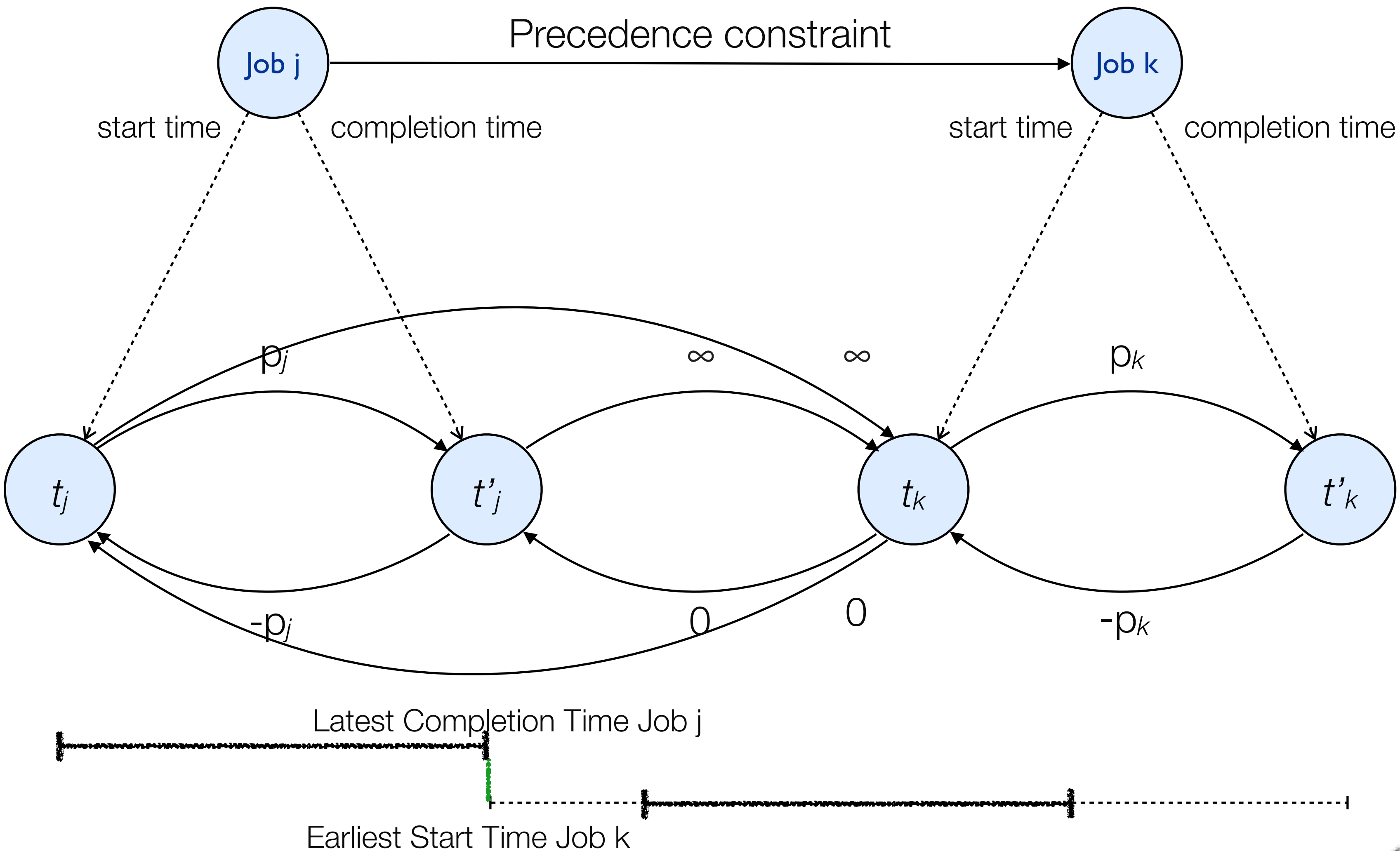
Key problem:
Job j and k may coincide; that is, the allowed time intervals (partially) overlap.
AND
Sum of resource requirements exceeds resource availability.

Solution (given the resource availability):
Add precedence constraint(s) to the existing set of constraints
such that no value assignment can cause a resource peak.

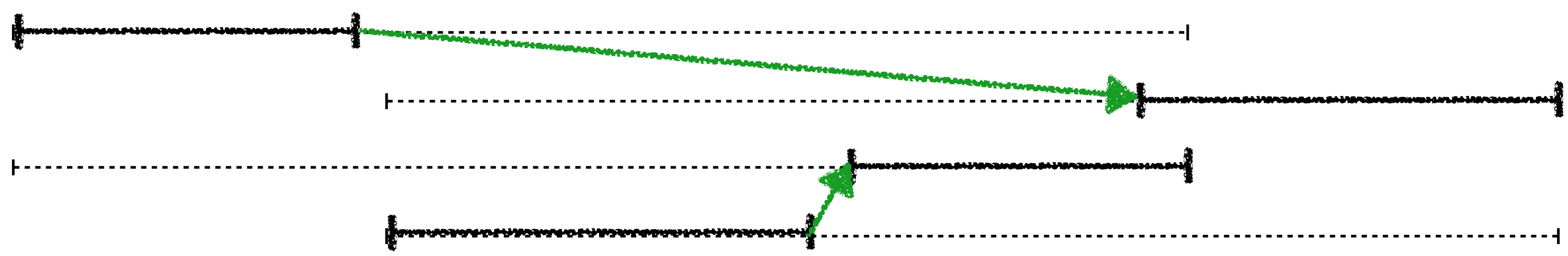
Constraint Posting

Resource availability: $R_1 = 1$

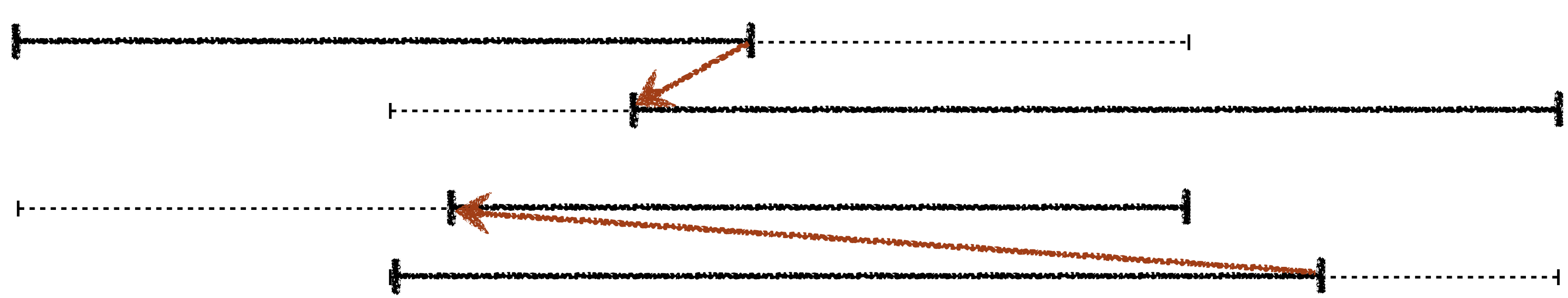
Resource requirements: $R_{1j} = 1$ $R_{1k} = 1$



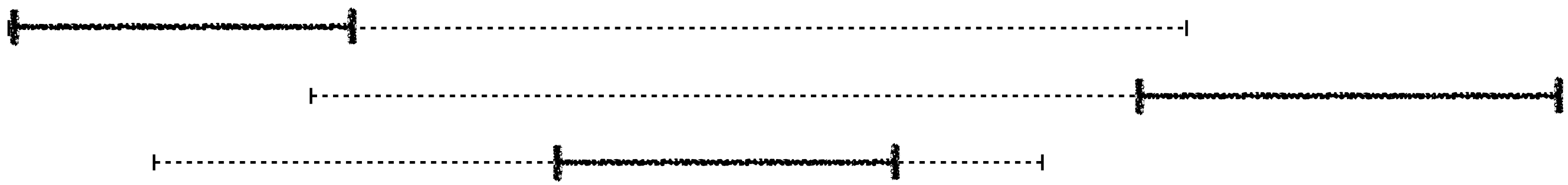
Precedence Constraint Options



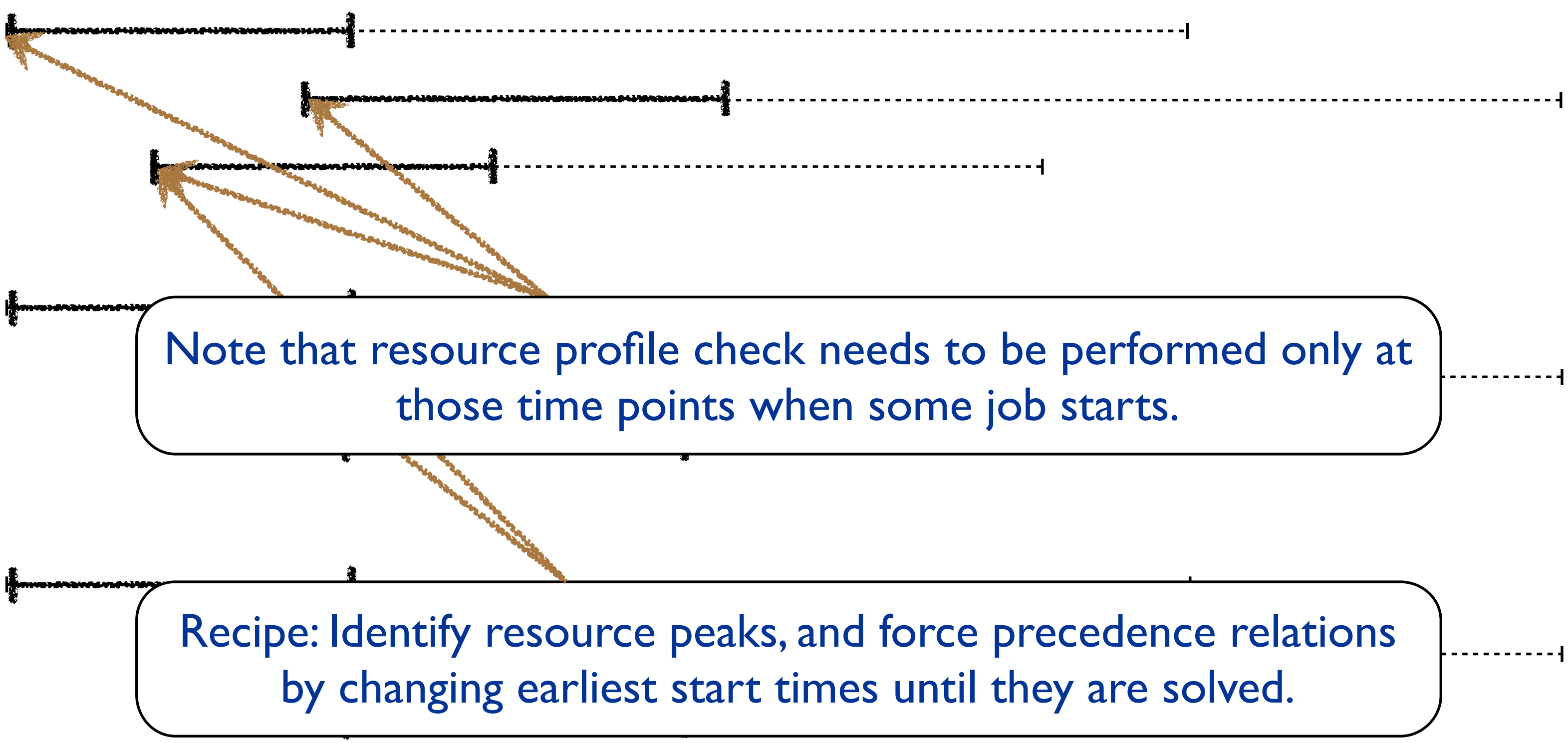
If
Earliest Start Time Job j + Processing Time Job j ≤ Latest Start Time Job k
then
Job j can proceed Job k.



Earliest Start Time Assumption



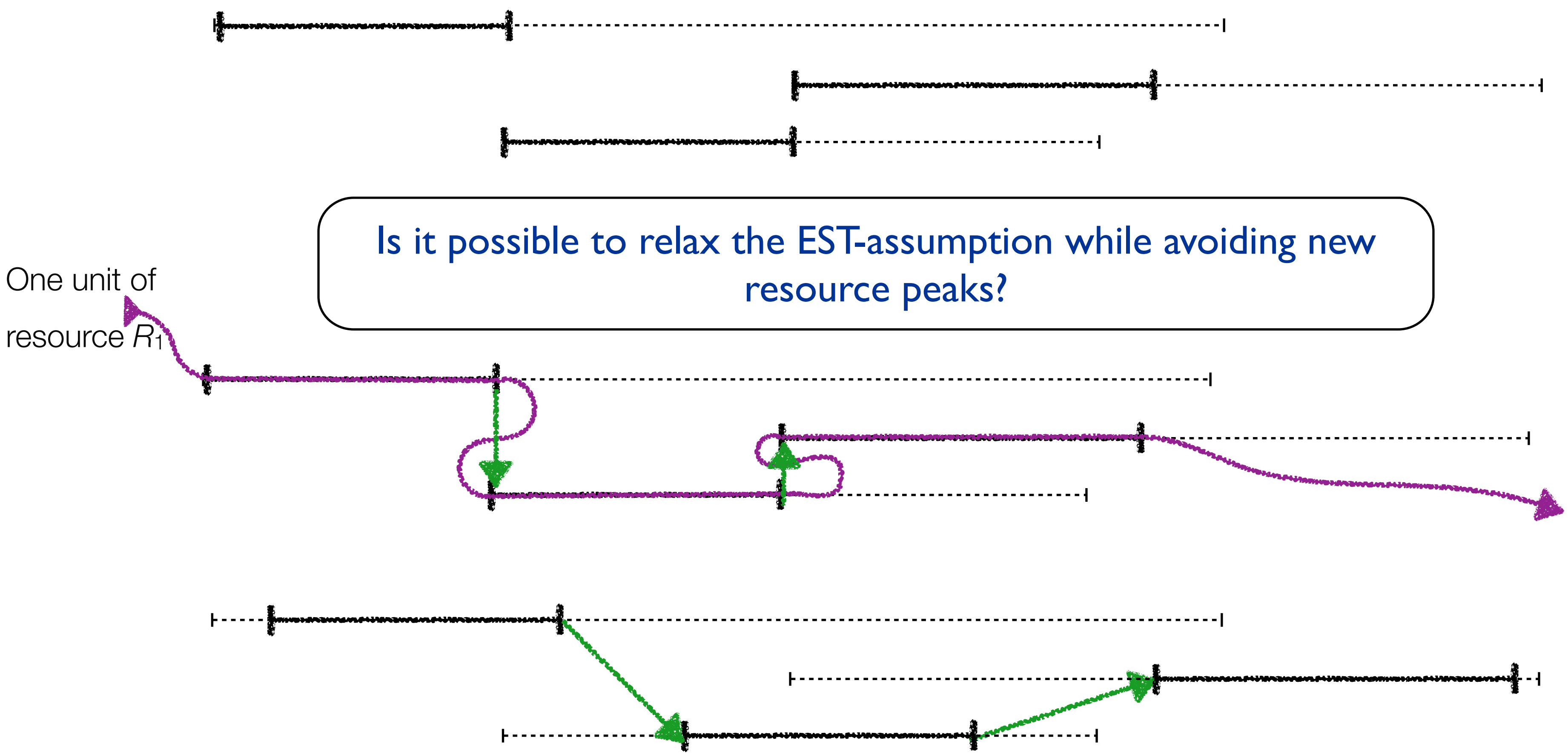
Approach: Assume that each job starts as soon as possible.
(Earliest Start Time Schedule)



Note that resource profile check needs to be performed only at those time points when some job starts.

Recipe: Identify resource peaks, and force precedence relations by changing earliest start times until they are solved.

Resource Chaining



Is it possible to relax the EST-assumption while avoiding new resource peaks?

Recipe: Identify a feasible job-to-job flows of resource units, and add precedence constraints accordingly.

Basics of some heuristic

- Find resource feasible EST Schedule
  - Assume jobs start as soon as possible
  - Repeat until all resource conflicts are resolved
    - Identify existing resource conflicts
    - Select resource for which activities should be scheduled first
    - Identify sequence options between competing activities
    - Calculate best activity sequence
    - Update earliest start times
- Identify feasible job-to-job resource flows (Chaining)
- Add precedence constraints according resource flows

Temporal Decoupling

Solving the independent scheduling problem in STP’s

Temporal decoupling

In finding solutions, until now we assumed that *one* team/actor controls the assignment of values to *all* the variables.

Suppose
- there is more than one team
- every team A_i controls a disjoint subset T_i of variables
- each team A_i wants to find a local solution to the sub-STN S_i generated by its set of time points T_i,

This is the **distributed scheduling problem** for STN's.

We discuss an efficient solution to this problem:

Temporal Decoupling

Temporal decoupling: example

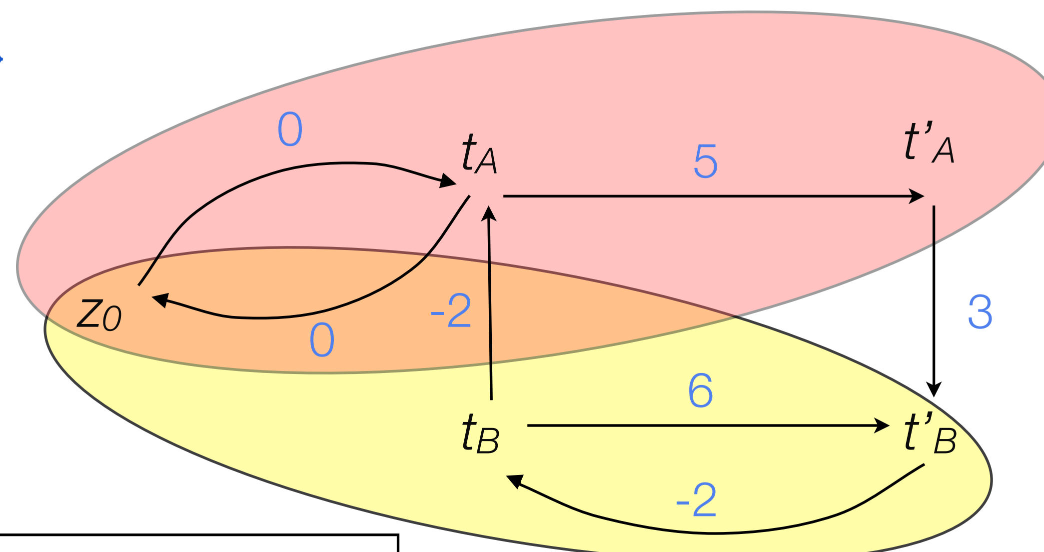
Consider the maintenance example discussed before.
Suppose there are two teams.
One team (A) has to job A, the other (B) job B.
They would like to determine their schedule independently from each other.

Suppose:

A chooses $Sol(S_A) \ni \{ t_A = 0, t'_A = 2 \}$

B chooses $Sol(S_B) \ni \{ t_B = 3, t'_B = 6 \}$

controlled by A



controlled by B

Both solutions satisfy the local constraints

But $Sol(S_A) \cup Sol(S_B)$ =
$\{ t_A = 0, t'_A = 2 , t_B = 3, t'_B = 6 \}$
is *not* a solution of S !

intuitive reason for failure:
some constraints are not implied by local constraints !

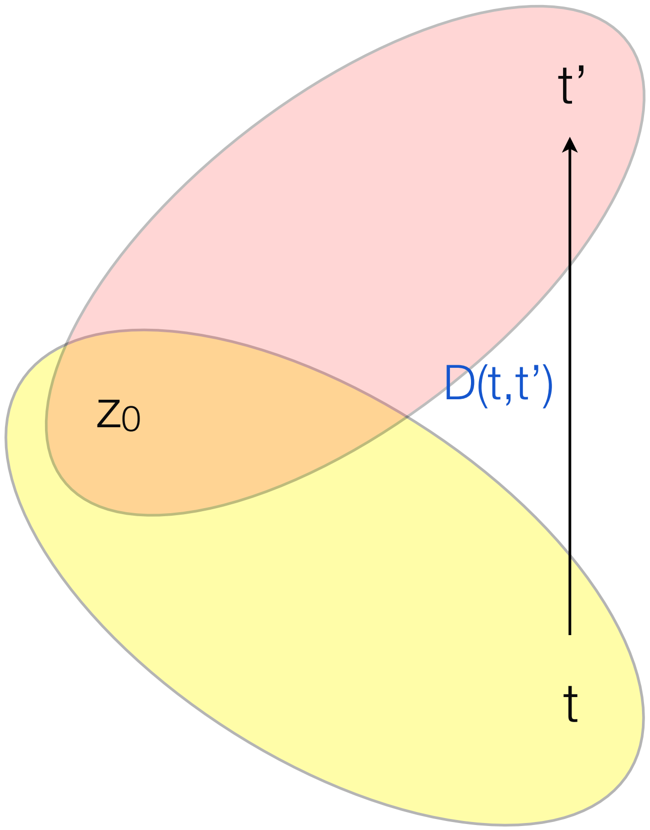
Temporal Decoupling:

the method

Temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.

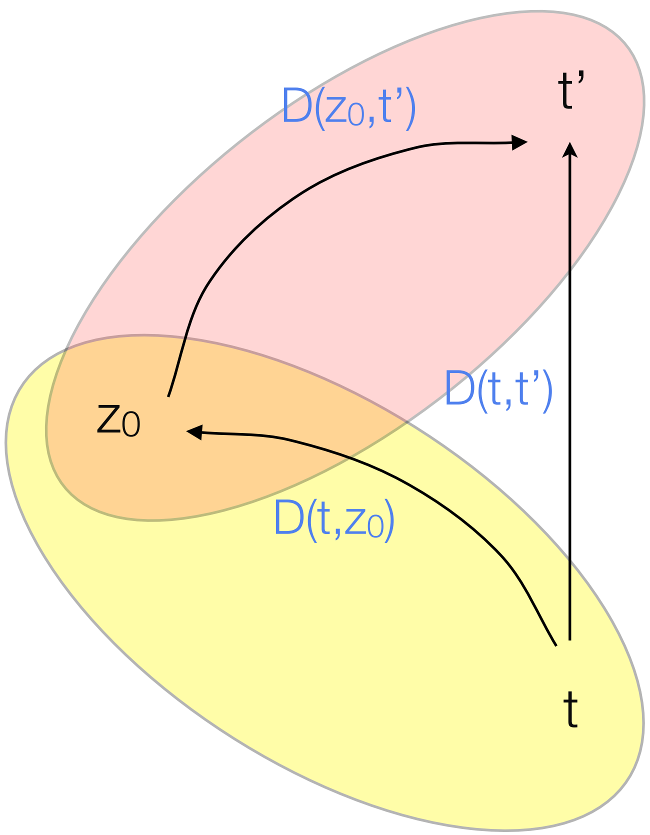
Consider the intra-team constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



Temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.

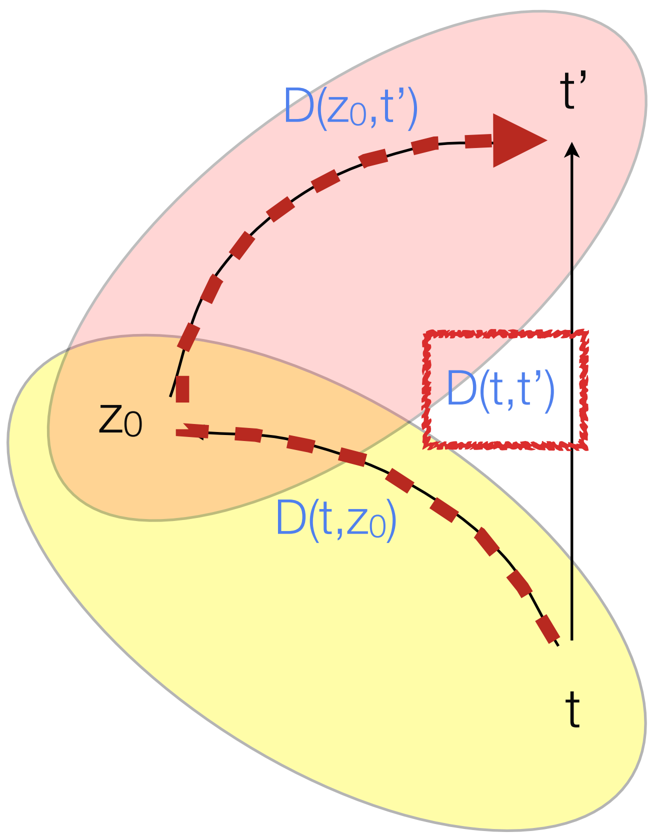
Consider the intra-team constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



Temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.

Consider the intra-team constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



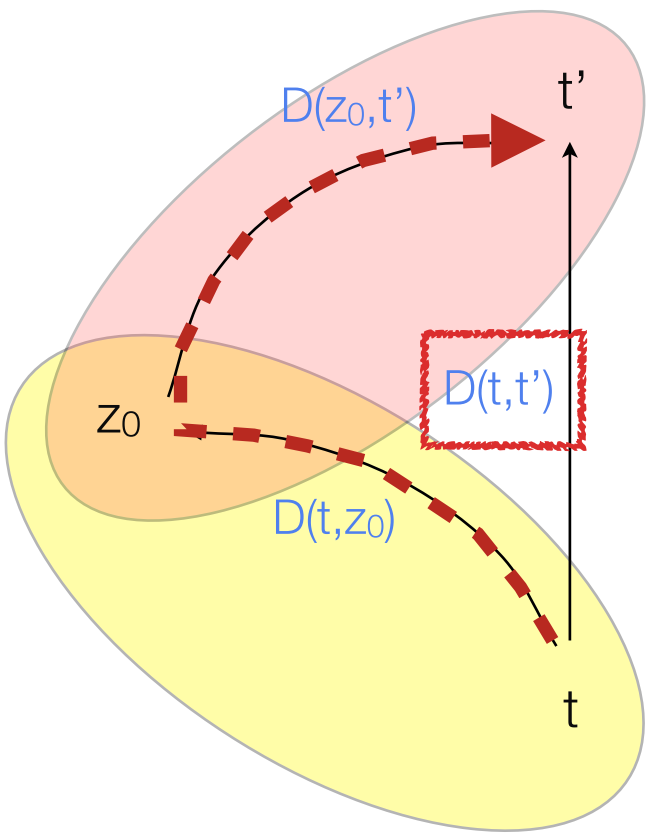
If $D(t,z_0) + D(z_0,t') > D(t,t')$ then $t' - t \leq D(t,t')$

is *not implied* by $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$

Temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.

Consider the intra-team constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



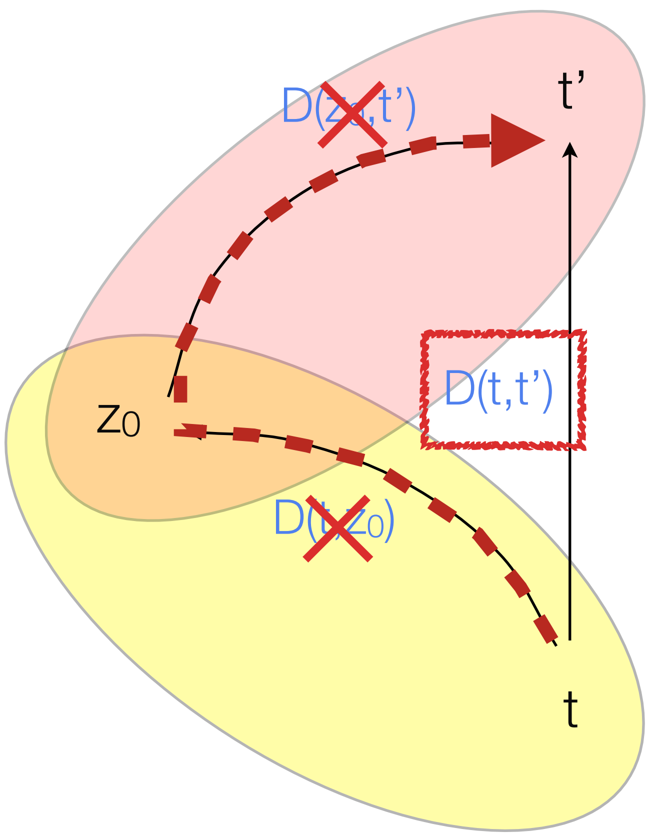
If $D(t,z_0) + D(z_0,t') > D(t,t')$ then $t' - t \leq D(t,t')$ is *not implied* by $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$

We can ensure implication of $t' - t \leq D(t,t')$ by intra-team constraints by *tightening* $t' - z_0 \leq D(z_0,t')$ and $z_0 - t \leq D(t,z_0)$

Temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.

Consider the intra-team constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



If $D(t,z_0) + D(z_0,t') > D(t,t')$ then $t' - t \leq D(t,t')$ is *not implied* by $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$

We can ensure implication of $t' - t \leq D(t,t')$ by intra-team constraints by *tightening* $t' - z_0 \leq D(z_0,t')$ and $z_0 - t \leq D(t,z_0)$

Method: Choose δ_t and $\delta_{t'}$ such that

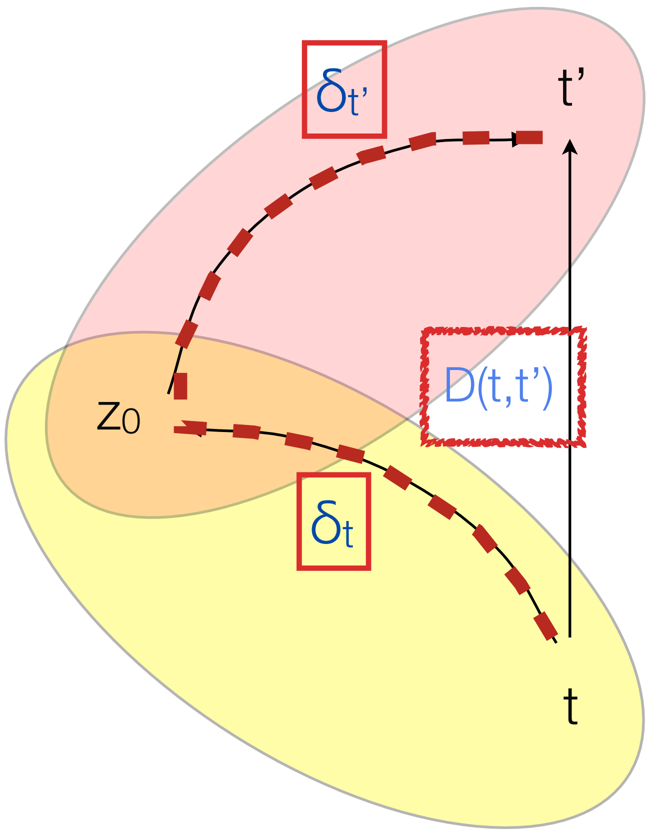
1. $-D(z_0,t) \leq \delta_t < D(t,z_0)$
2. $-D(t,z_0) \leq \delta_{t'} < D(z_0,t')$
3. $\delta_t + \delta_{t'} \leq D(t,t')$

Add constraints $t' - z_0 \leq \delta_{t'}$ and $z_0 - t \leq \delta_t$ to S and compute new distance matrix.

Temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.

Consider the intra-team constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



$t' - t \leq D(t,t')$ implied by $t' - z_0 \leq \delta_{t'}$ and $z_0 - t \leq \delta_t$

If $D(t,z_0) + D(z_0,t') > D(t,t')$ then $t' - t \leq D(t,t')$ is *not implied* by $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$

We can ensure implication of $t' - t \leq D(t,t')$ by intra-team constraints by *tightening* $t' - z_0 \leq D(z_0,t')$ and $z_0 - t \leq D(t,z_0)$

Method: Choose δ_t and $\delta_{t'}$ such that

1. $-D(z_0,t) \leq \delta_t < D(t,z_0)$
2. $-D(t,z_0) \leq \delta_{t'} < D(z_0,t')$
3. $\delta_t + \delta_{t'} \leq D(t,t')$

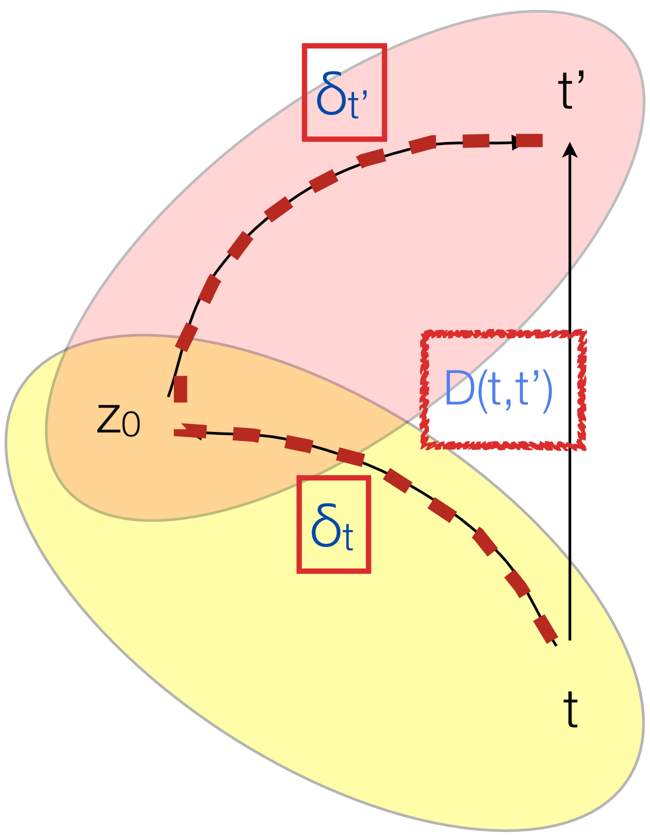
Add constraints $t' - z_0 \leq \delta_{t'}$ and $z_0 - t \leq \delta_t$ to S and compute new distance matrix.

TU Delft

Temporal decoupling: method

Take an arbitrary constraint $t' - t \leq D(t,t')$ such that t and t' belong to different blocks.

Consider the intra-team constraints $t' - z_0 \leq D(z_0,t')$, $z_0 - t \leq D(t,z_0)$.



If $t' - t \leq D(t,t')$ is *implied* by $t' - z_0 \leq \delta_{t'}$ and $z_0 - t \leq \delta_t$, it can be removed from S without any consequence.

This procedure can be repeated for every inter team constraint not implied by intra-team constraints.

The resulting system is a **decoupled STN**

Summary

- We discussed the RCPSP scheduling problem specifying both time and resource constraints.
  RCPSP can be used to model maintenance scheduling problems.

- There is a method to convert the hard to solve RCPSP to a similar easy to solve STP problem containing time constraints only.

- The method to solve STP problems provides a set of solutions, from which an arbitrary solution can be chosen in an easy way. Hence flexibility in solution finding is guaranteed.

- If the problem has to be distributed over teams who want to schedule independently, there exists an efficient temporal decoupling method for STP's