Robust scheduling of task graphs under execution time uncertainty

Michele Lombardi

Michela Milano

Luca Benini

Talk topic

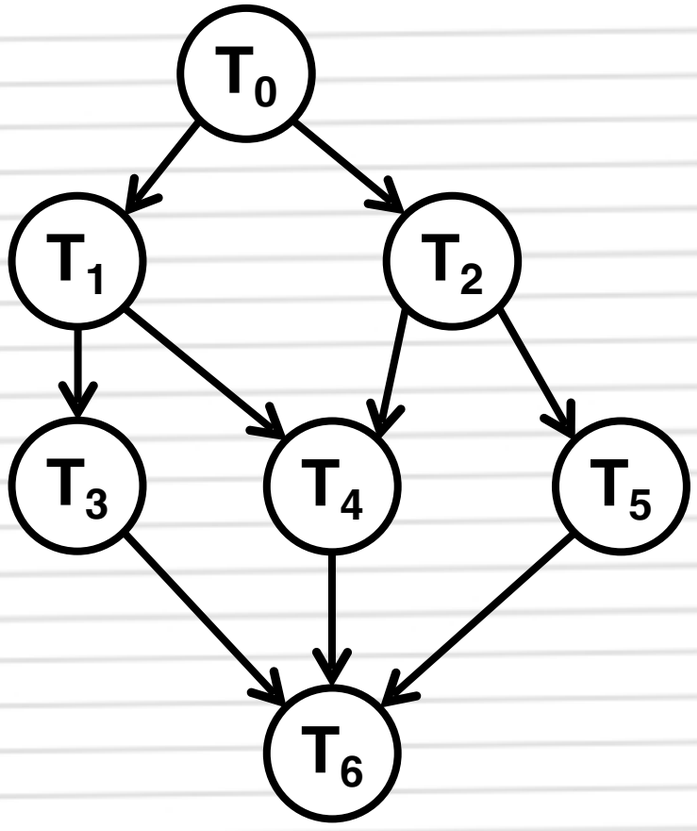
Main topic: reliable scheduling of multitask applications on multiprocessor systems in the presence of execution time uncertainty

Some assumptions:

- Tasks have temporal dependencies
- No preemption support
- Hard deadline constraints are specified

Context:

- Real time applications
- Clustered, domain specific multicore platforms (e.g. Cell, Nomadik...)

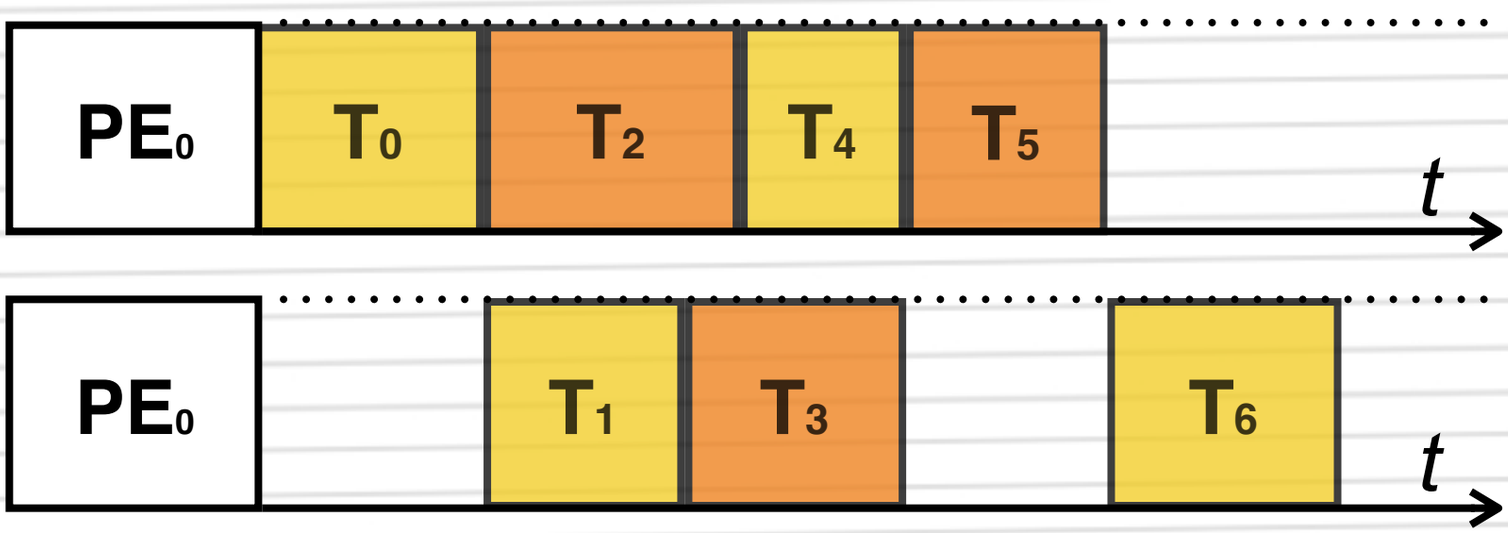


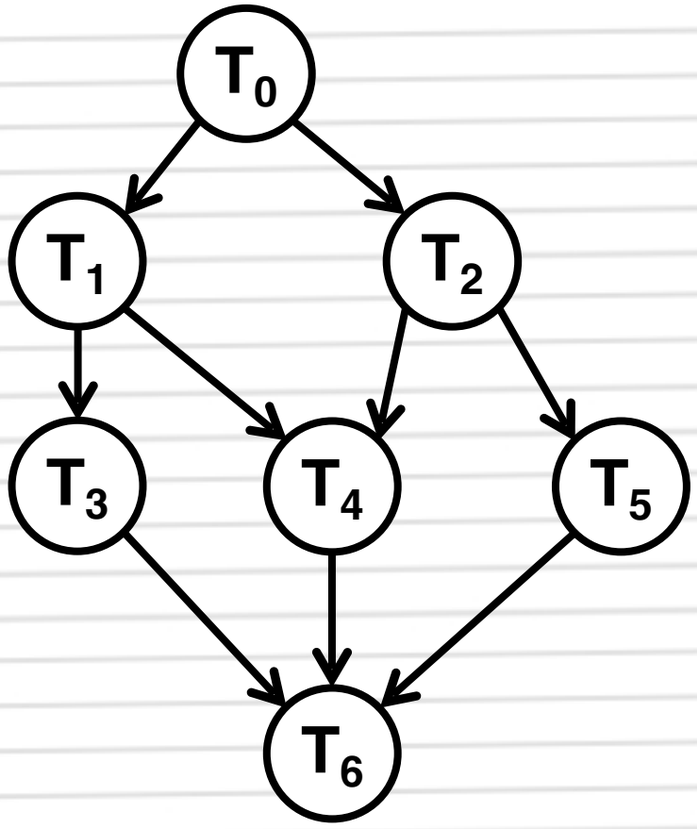
Off-line allocation and scheduling

Off-line allocation and scheduling, PROs and CONs:

- ↑ Improves application performance
- ↓ Requires a lot of a-priority knowledge
- ↓ Requires conservative assumptions (e.g. force WCET durations & force fixed start times to avoid anomalies)

A fixed start time, WCET schedule:



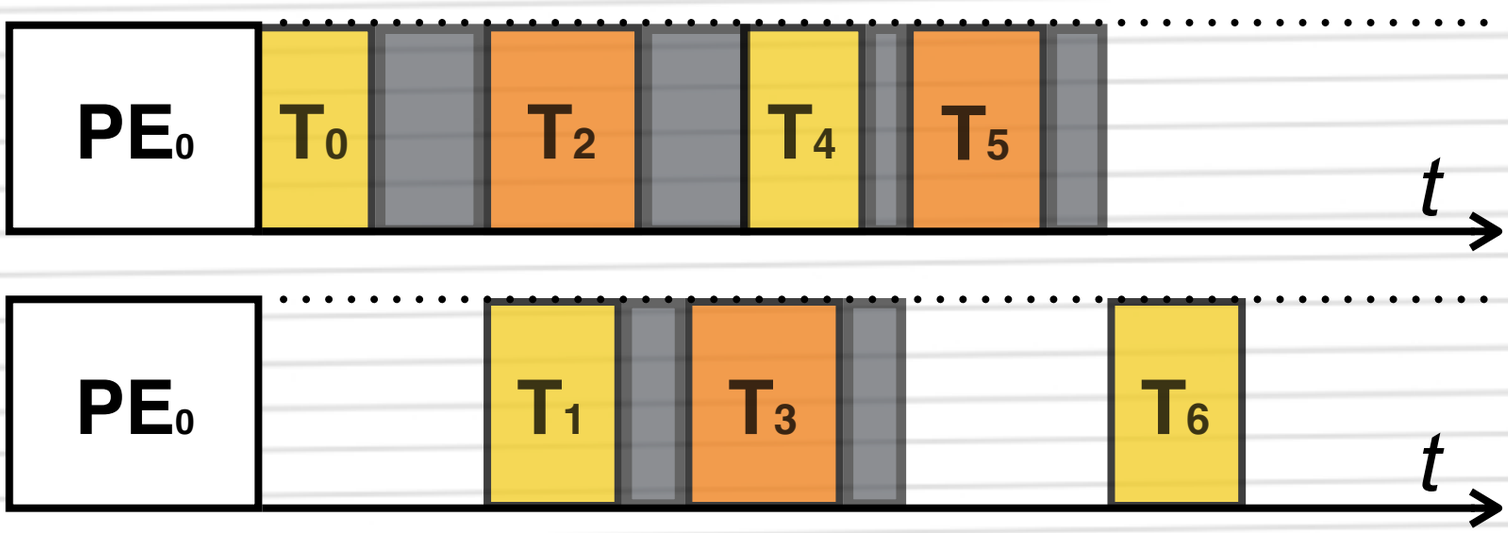


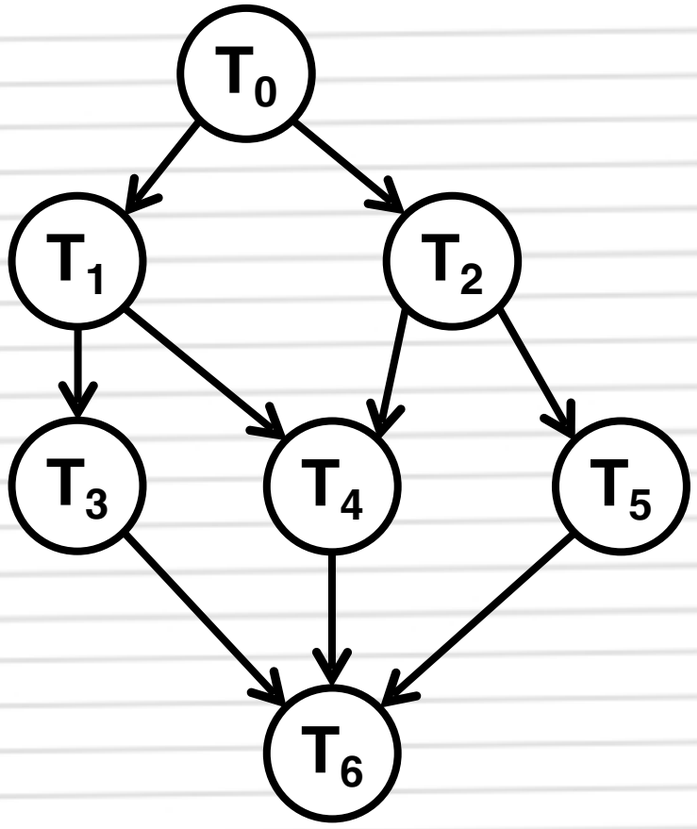
Off-line allocation and scheduling

Off-line allocation and scheduling, PROs and CONs:

- ↑ Improves application performance
- ↓ Requires a lot of a-priority knowledge
- ↓ Requires conservative assumptions (e.g. force WCET durations & force fixed start times to avoid anomalies)

A fixed start time, WCET schedule:



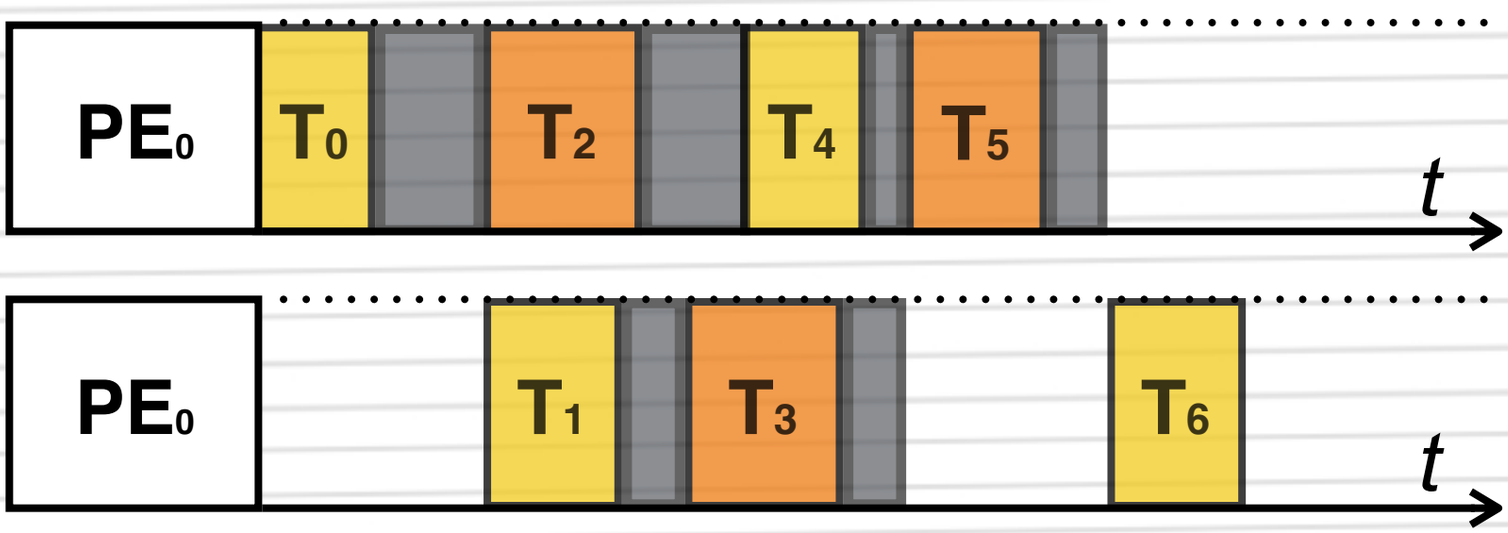


Off-line allocation and scheduling

Off-line allocation and scheduling, PROs and CONs:

- ↑ Improves application performance
- ↓ Requires a lot of a-priority knowledge
- ↓ Requires conservative assumptions (e.g. force WCET durations & force fixed start times to avoid anomalies)

A fixed start time, WCET schedule:



IDLE Time insertion

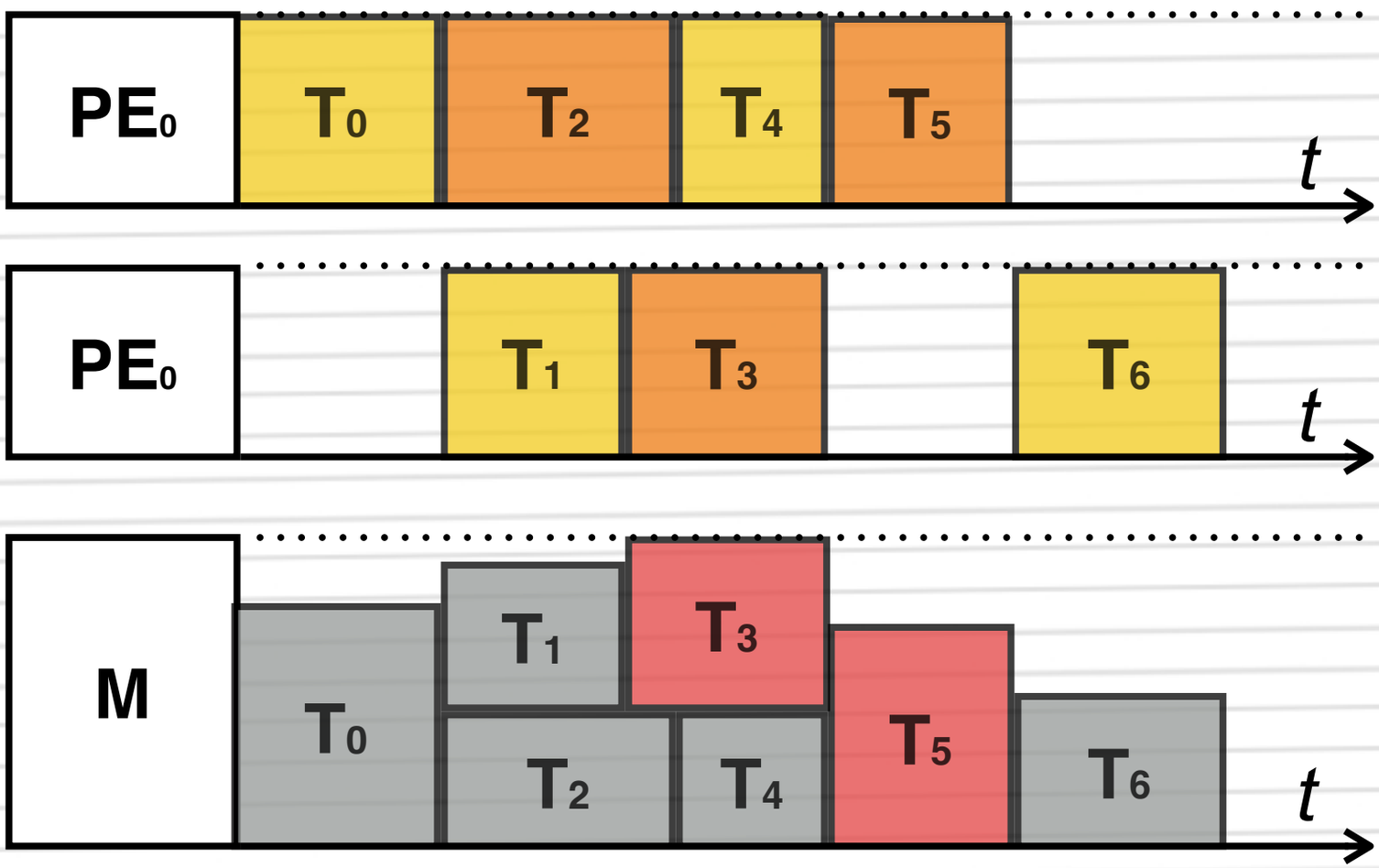
Timers and interrupts

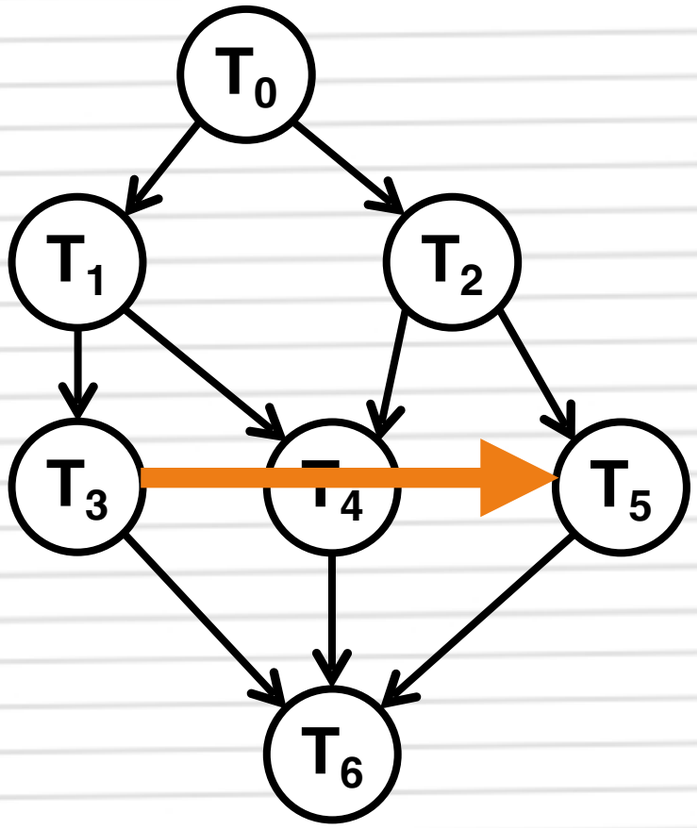
Precedence Constraint Posting

Precedence Constraint Posting

Provide a schedule as a set of **additional precedence relations**

- Those are added to avoid resource contention => **no anomalies**
- Tasks are started when all predecessors are over => **little idle time**



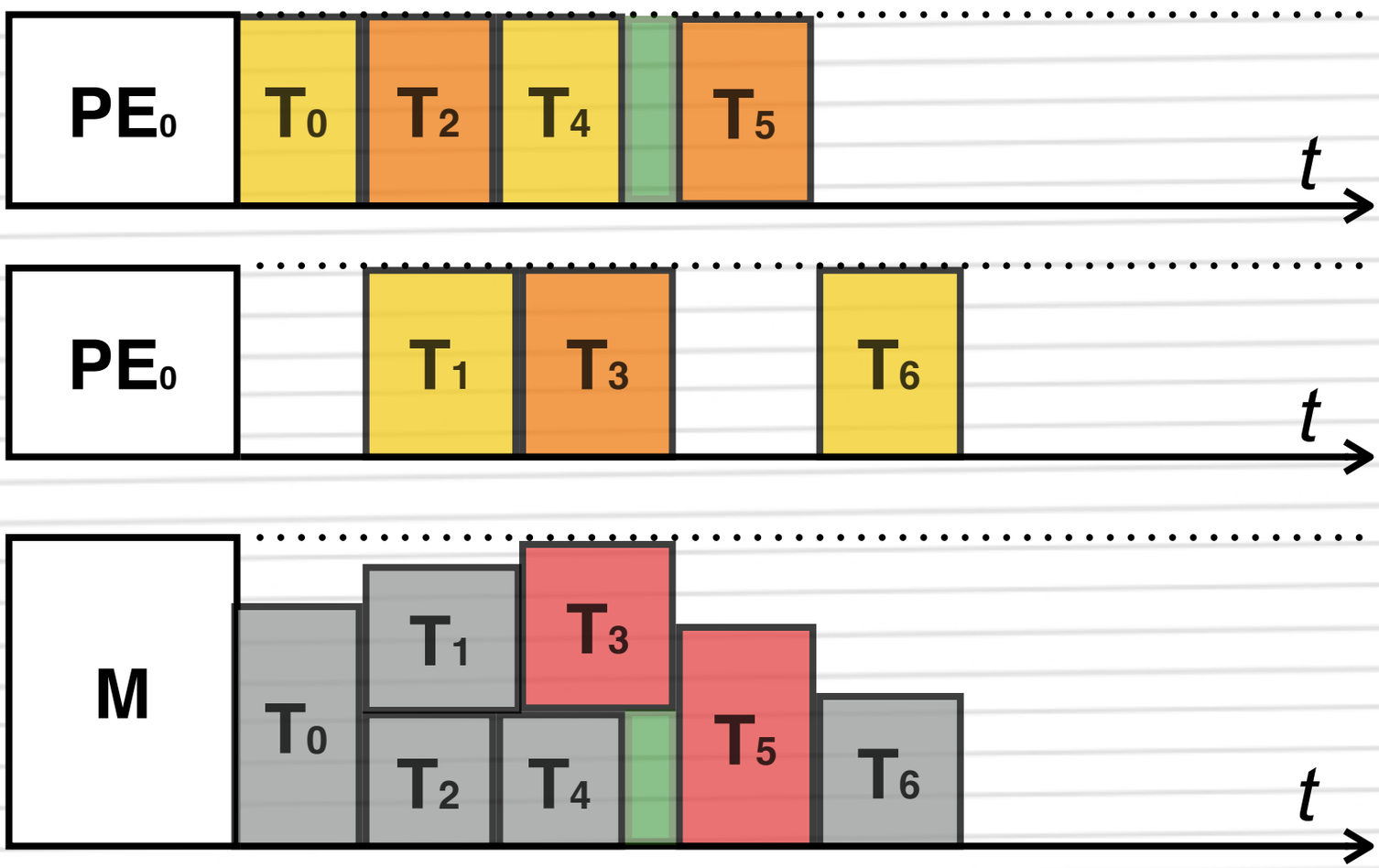


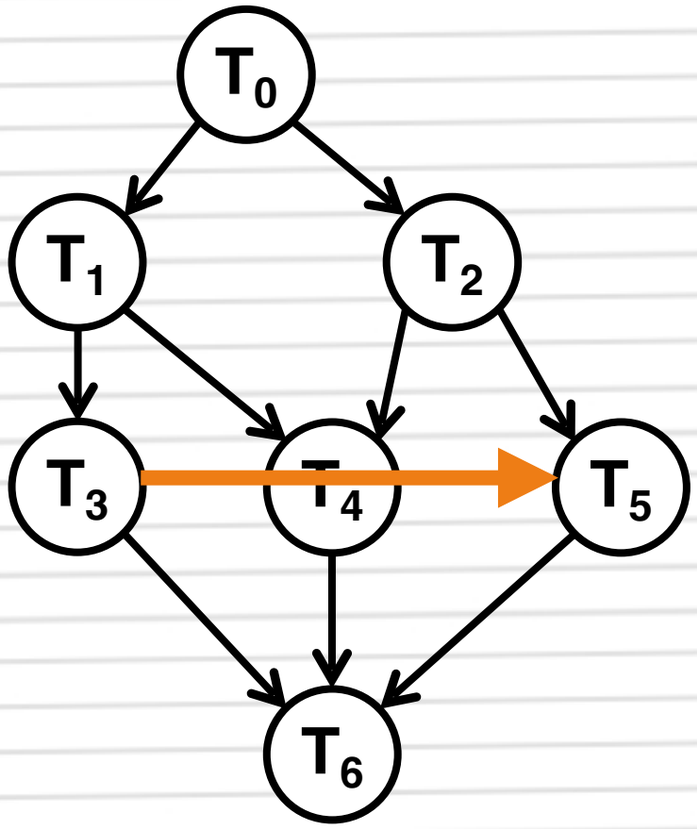
Precedence Constraint Posting

Precedence Constraint Posting

Provide a schedule as a set of **additional precedence relations**

- Those are avoid to avoid resource contention => **no anomalies**
- Tasks are started when all predecessors are over => **little idle time**





Precedence Constraint Posting

Precedence Constraint Posting is **a hybrid scheduling method:**

- A graph augmentation is computed off-line
- But it is executed by an on-line scheduler

In detail, the output of the on-line process is:

- A set of precedence relations
- A set of restricted time windows

The on-line scheduler must:

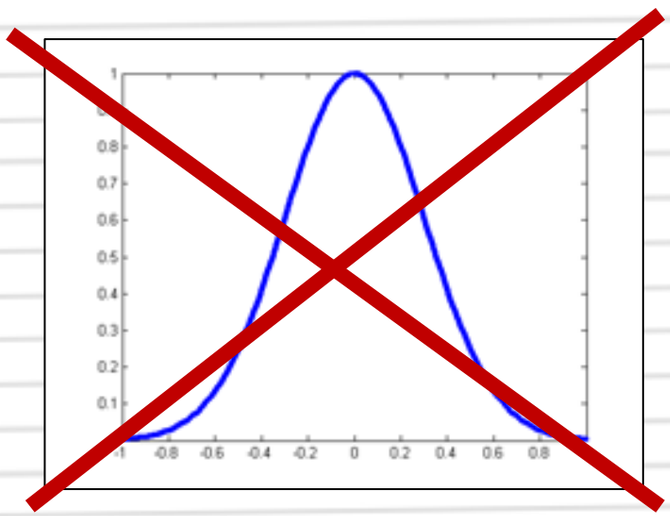
- Start a task when the predecessors are over
- Start a task within the restricted time window

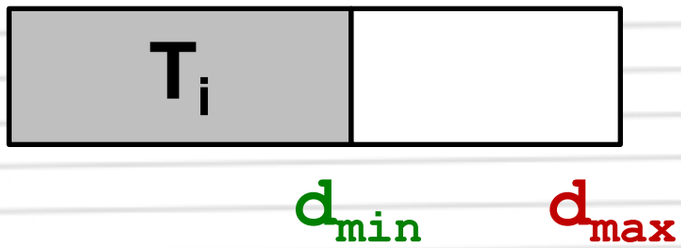
The on-line scheduler <u>can</u> be smarter than this

- E.g. a task as soon as possible

Characterization of uncertainty

- Preventing resource conflicts requires some knowledge on duration variability
- We use **duration bounds rather than probability distributions**

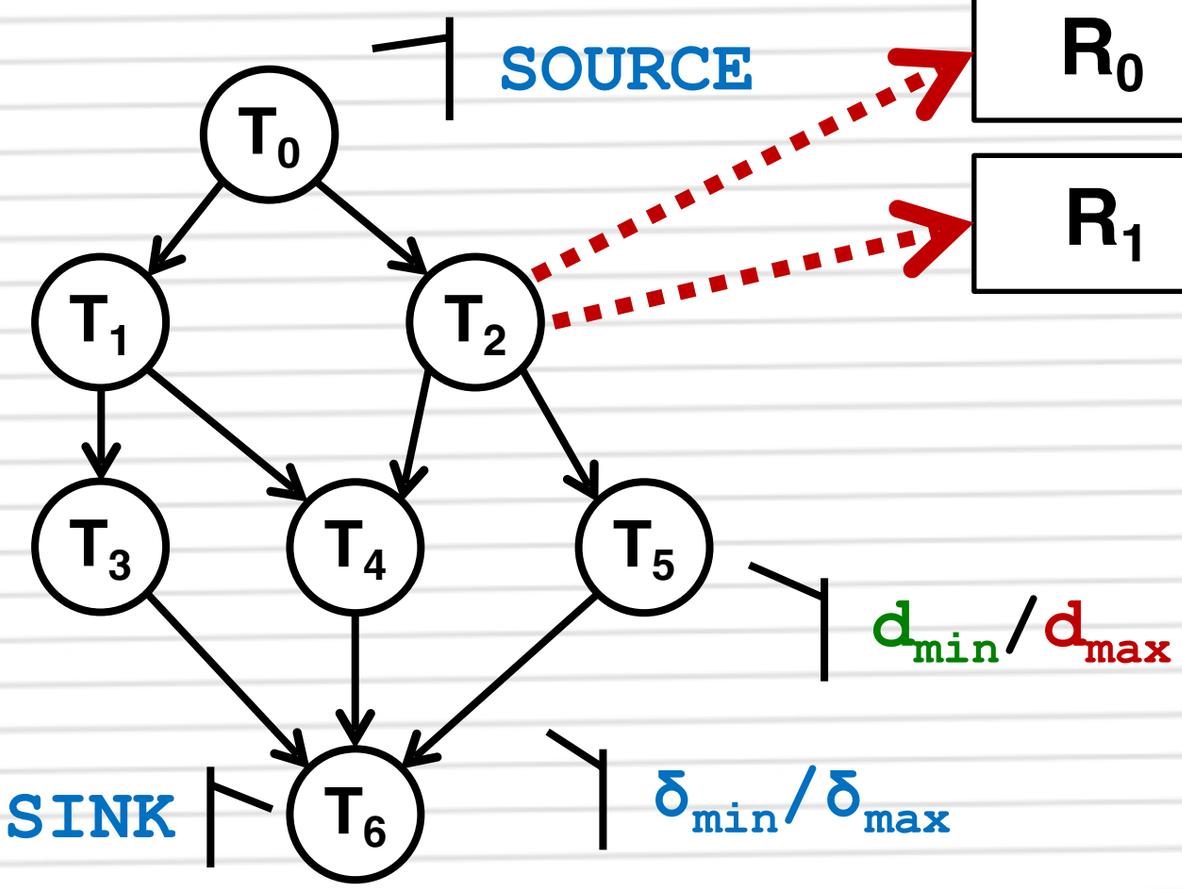




- Easier to obtain than probability distributions
- More computationally tractable
- Poor expected completion time approximation

The target problem

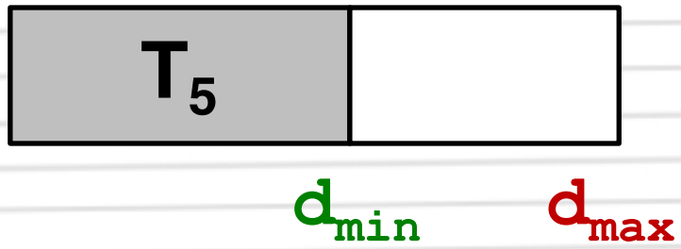
Task Graph



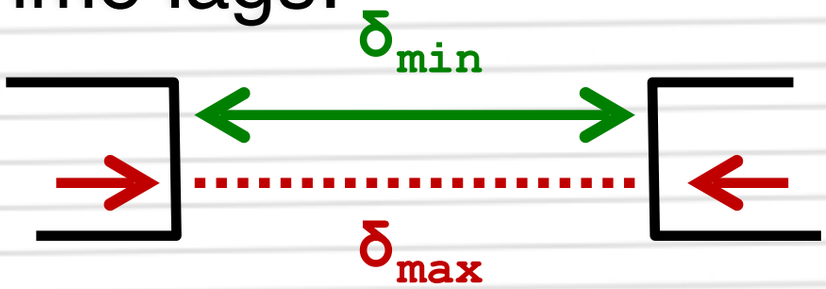
Resources:

	C	T0	T1	T2	T3	T4	T5	T6
R_0	2	--	2	--	1	2	1	--
R_1	3	--	1	2	--	2	1	--
d_{min}		0	1	2	1	1	1	0
d_{max}		0	2	3	3	2	2	0

Variable durations:



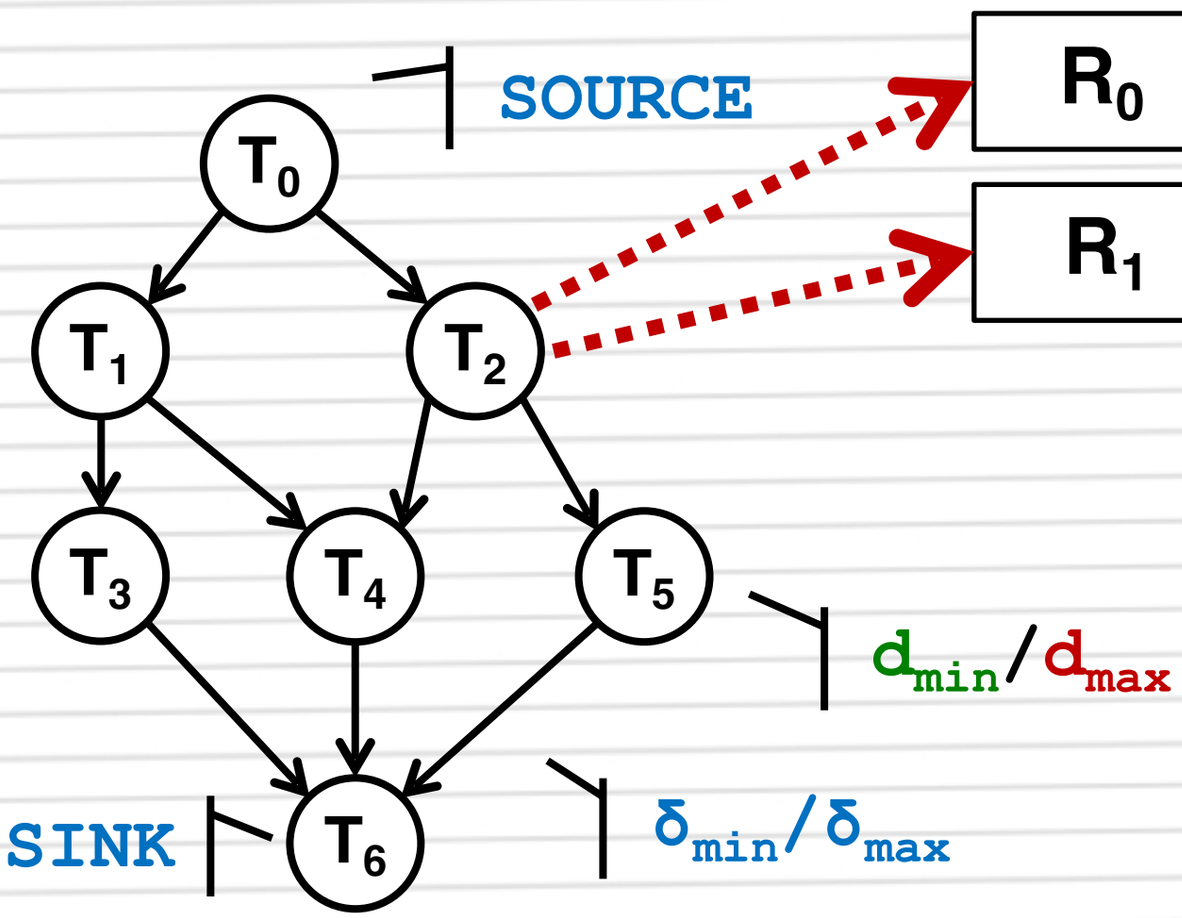
Time lags:



- Setup times
- max-latency csts

The target problem

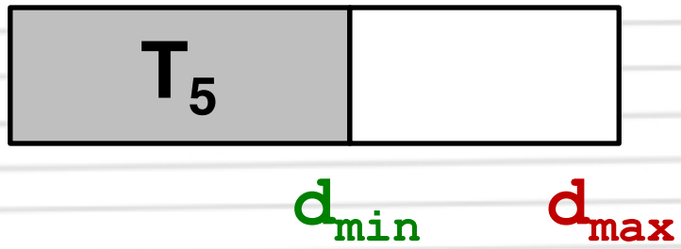
Task Graph



Resources:

	C	T0	T1	T2	T3	T4	T5	T6
R_0	2	--	2	--	1	2	1	--
R_1	3	--	1	2	--	2	1	--
d_{min}		0	1	2	1	1	1	0
d_{max}		0	2	3	3	2	2	0

Variable durations:



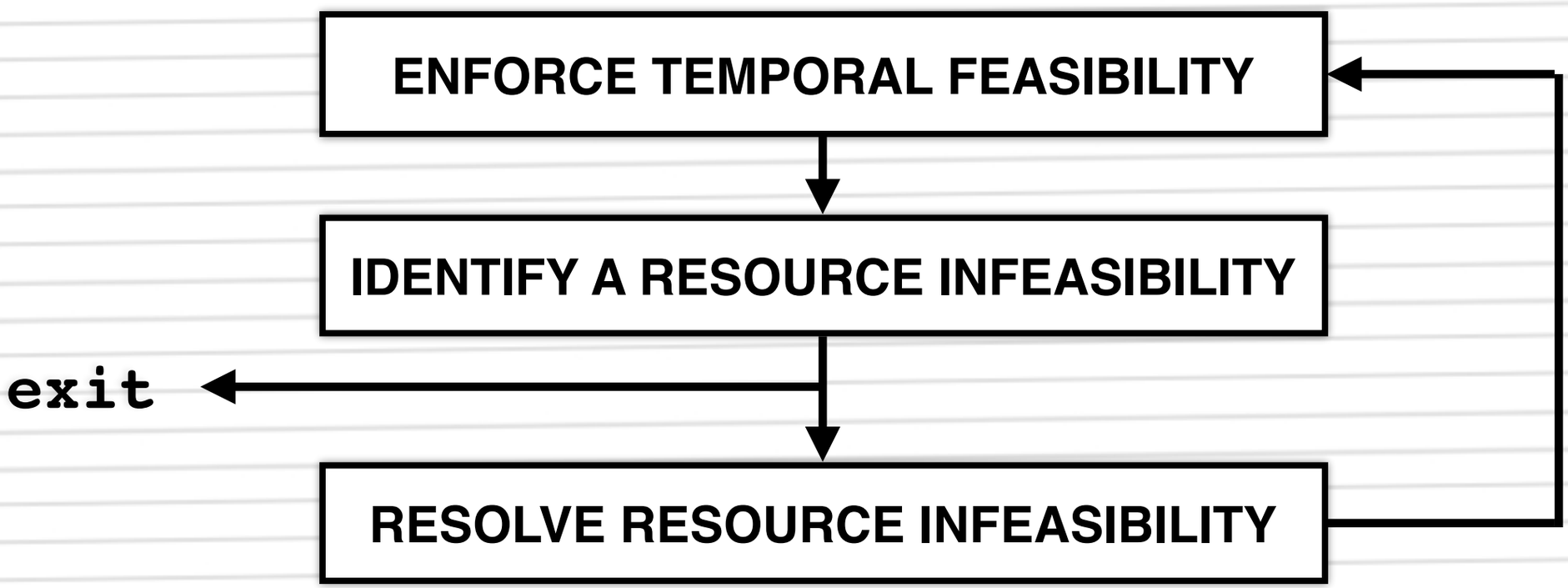
- Deadline and release times can be specified for each task
- We assume a resource allocation is provided by an external tool

Scheme of the solution approach

Base algorithm: **Depth First Search**

Each node represents a **partial schedule** (i.e. set of precedence relations and time windows):

- Temporally feasible
- Resource infeasible



Temporal Consistency

Issue #1: temporal consistency of the augmented network

Temporal model ⊆ STN with Uncertainty [Vidal,Fargier '99]

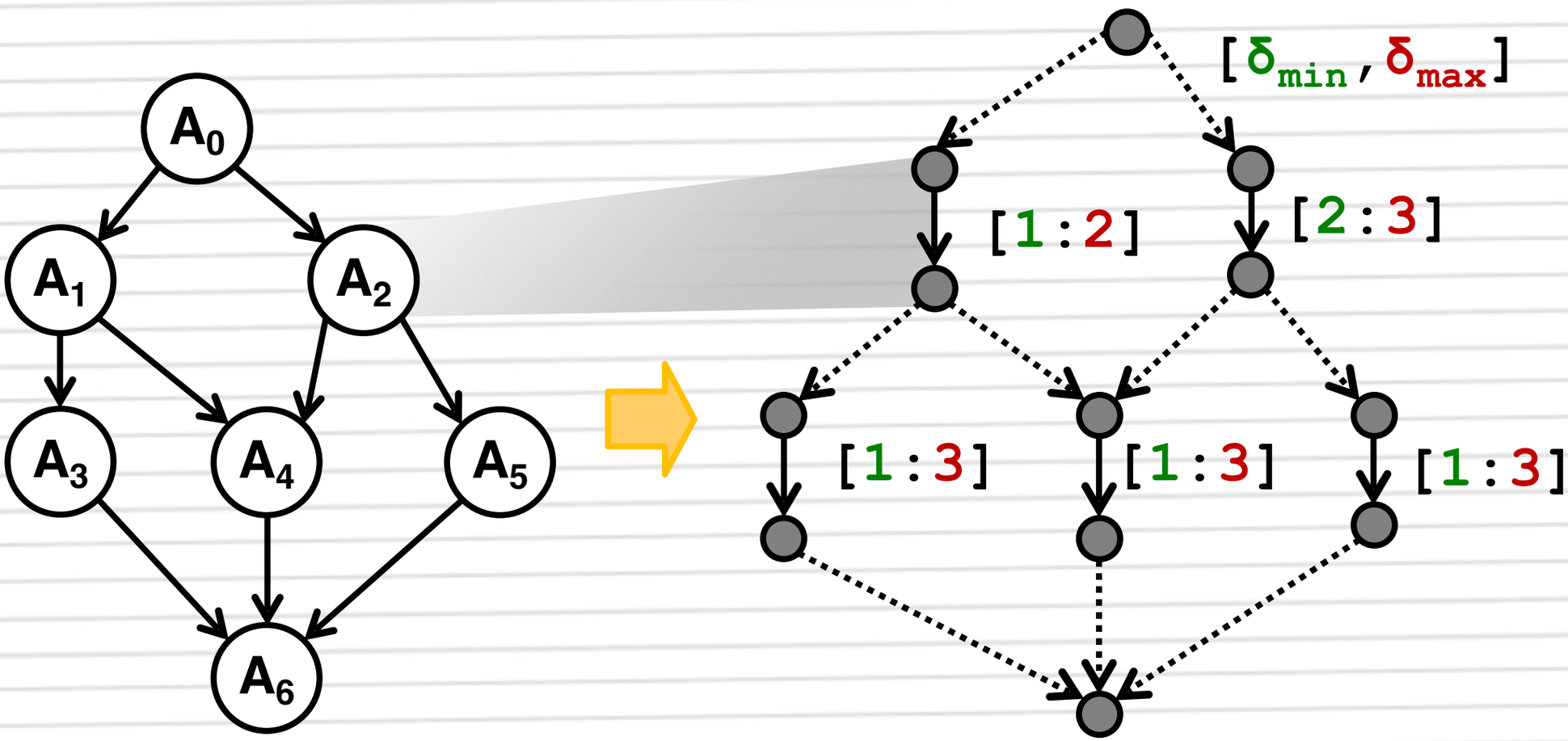
- Event variables:
- Free constraint: $\xrightarrow{[l,u]}$ $\exists d \in [l,u],\ l \geq 0$
- Contingent constraint: $\xrightarrow{[l:u]}$ $\forall d \in [l,u],\ l \geq 0$

Domain = time window

Temporal Consistency

Issue #1: temporal consistency of the augmented network

Temporal model ⊆ STN with Uncertainty [Vidal,Fargier '99]

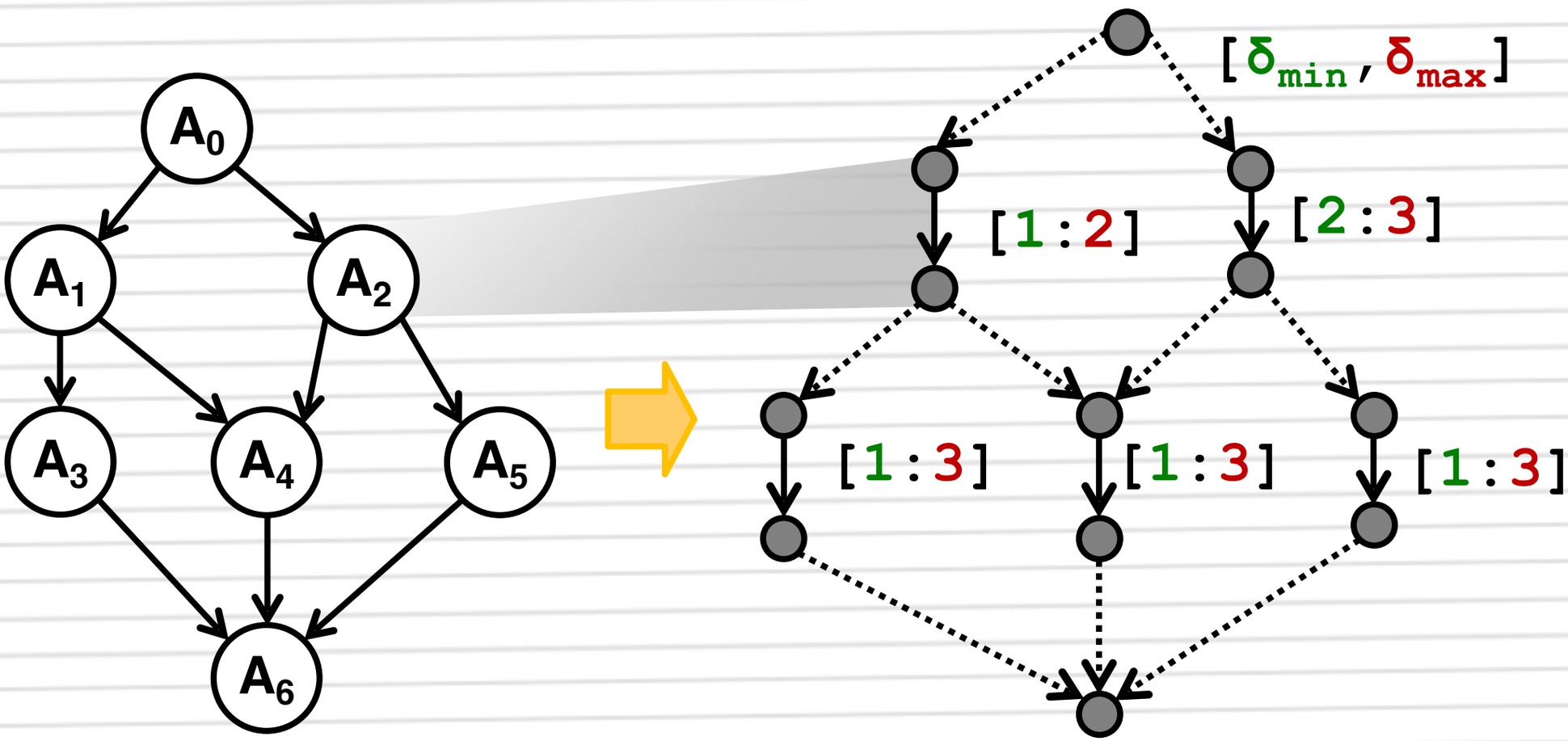


	d_{min}	d_{max}
A_0	0	0
A_1	1	2
A_2	2	3
A_3	1	3
A_4	1	2
A_5	1	2
A_6	0	0

Temporal Consistency

Constraint propagation:

- When the domain of an event variable is modified, filtering algorithms remove probably infeasible values from the domain of other event variables

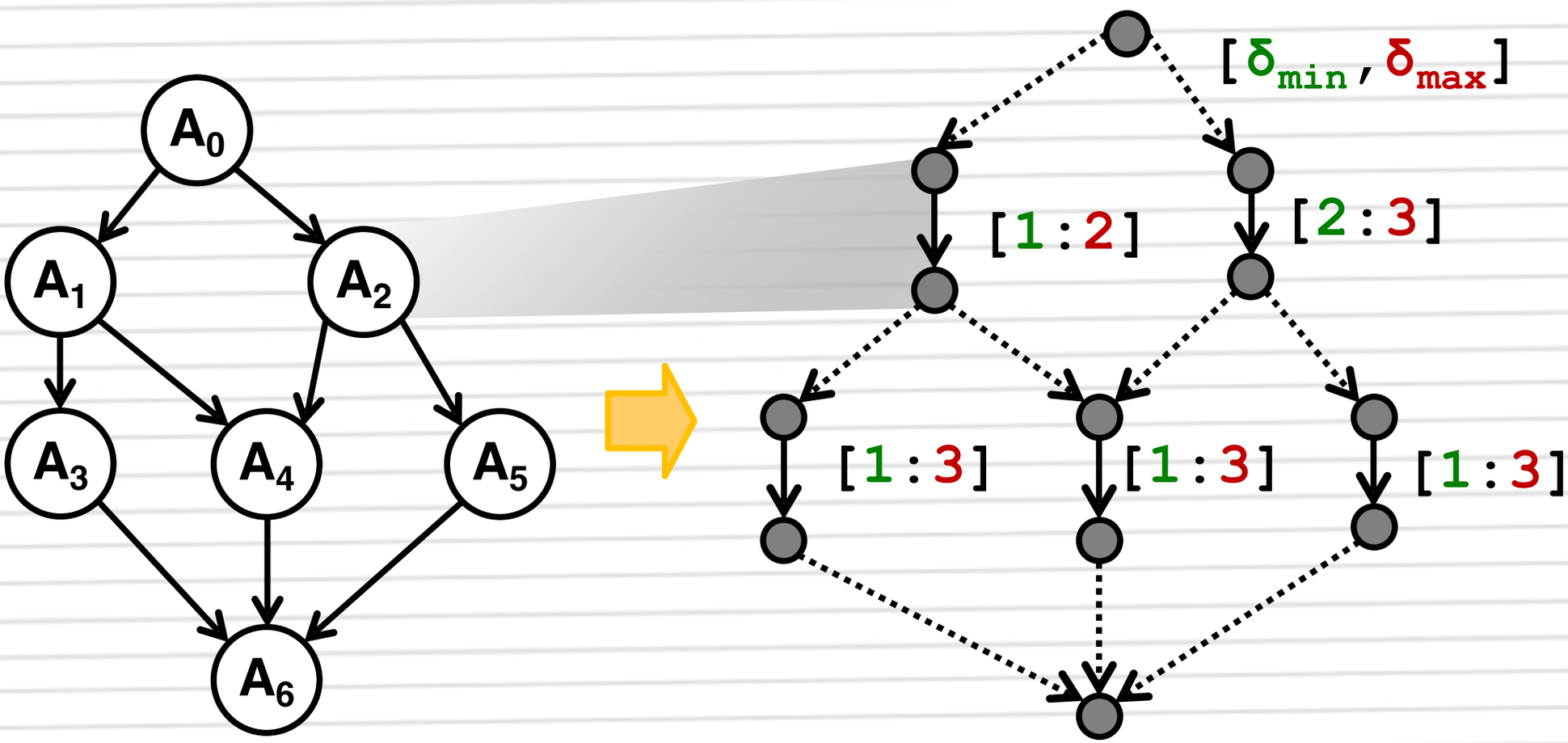


	d_{min}	d_{max}
A_0	0	0
A_1	1	2
A_2	2	3
A_3	1	3
A_4	1	2
A_5	1	2
A_6	0	0

Temporal Consistency

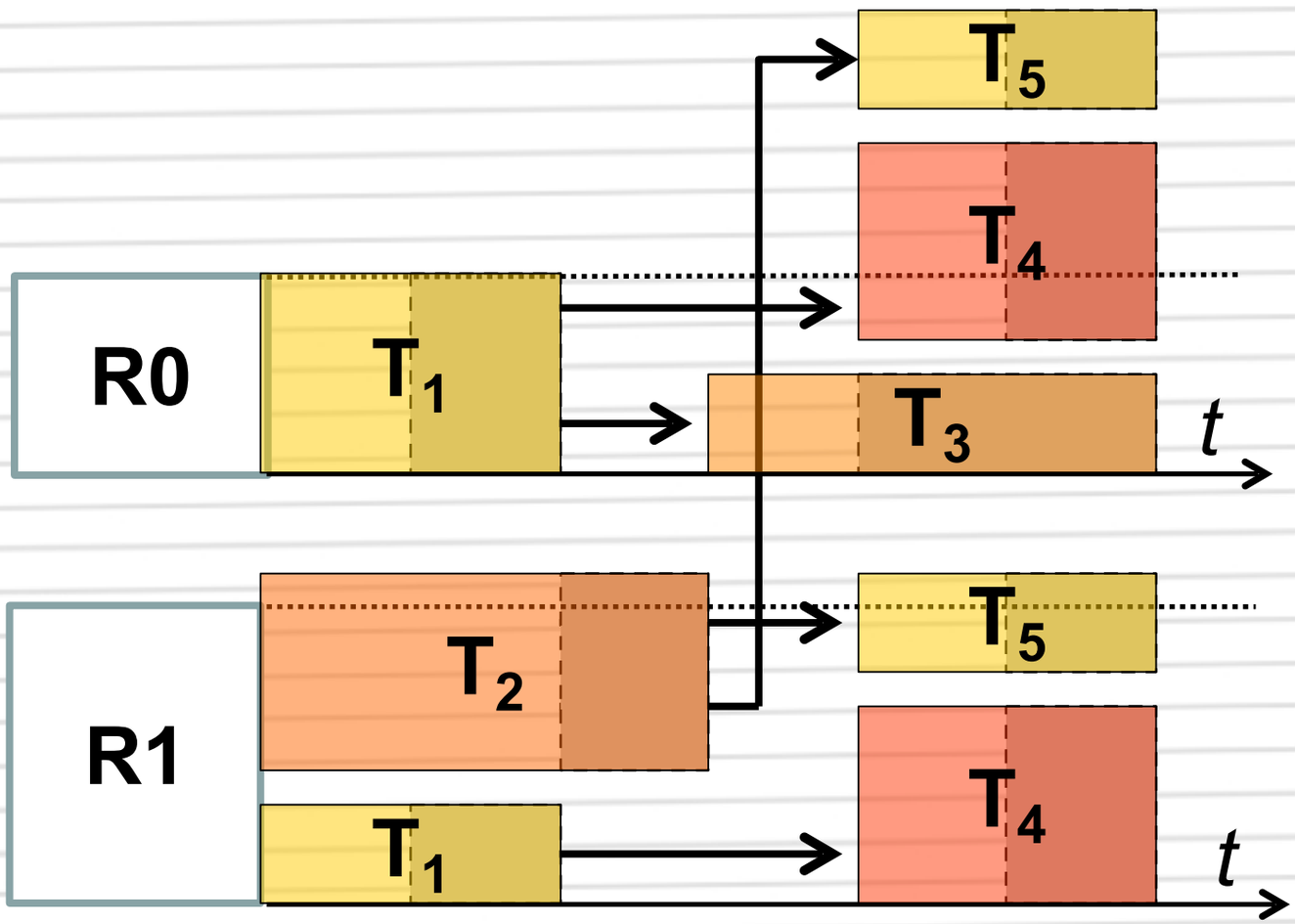
Constraint propagation:

- When any domain gets empty, the search process has met a dead-end and a backtrack occurs



	d_{min}	d_{max}
A_0	0	0
A_1	1	2
A_2	2	3
A_3	1	3
A_4	1	2
A_5	1	2
A_6	0	0

Resource consistency: Minimal Critical Sets



Critical Sets

$\{A_1,A_5\}$, ~~$\{A_3,A_4,A_5\}$~~,
$\{A_3,A_4\}$, $\{A_4,A_5\}$

A Minimal Critical Set is **resolved** by posting a new precedence cst.

No MCS = resource consistency

Minimal Critical Set = minimal set of possibly overlapping tasks, collectively overusing a resource

MCS Identification

Issue #2: efficiet MCS identification

the **MCSs are Independent Sets** on the augmented graph => in exponential number

=> **In principle**, **NP-complete** identification

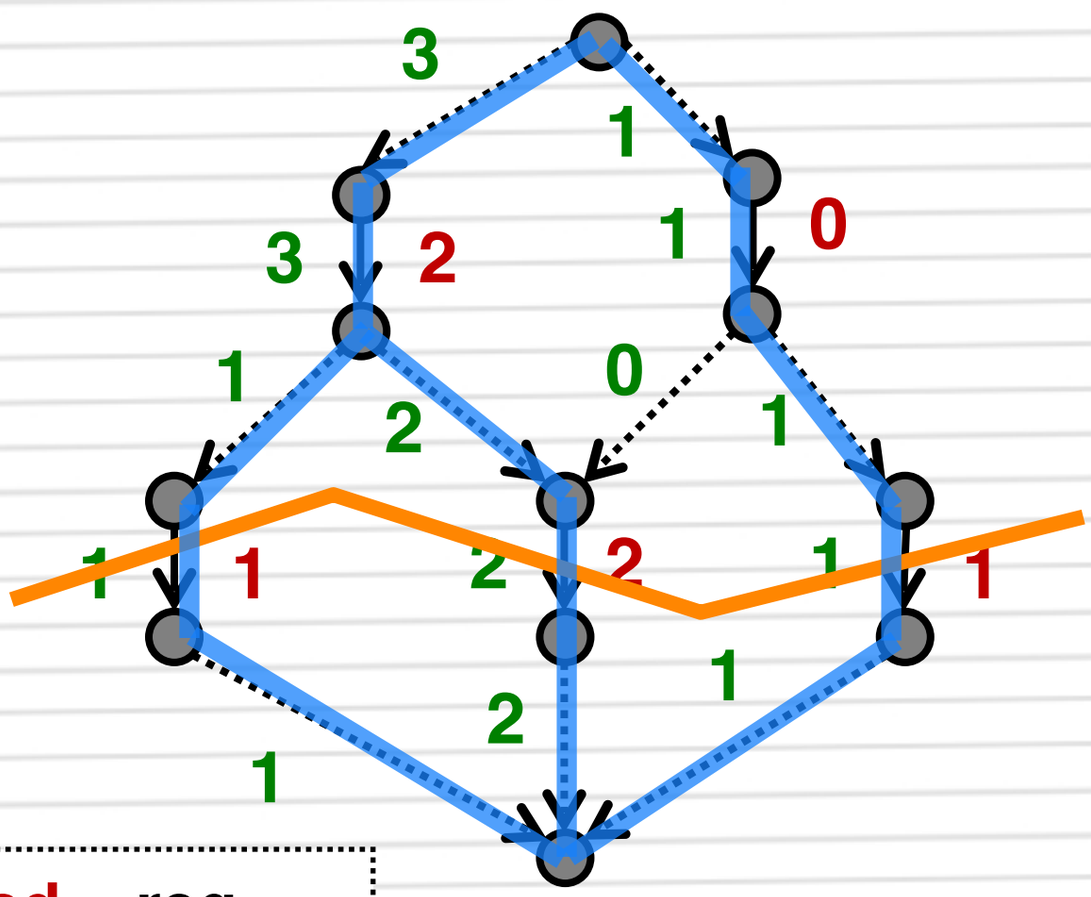
In practice, the graph has a transitive orientation

=> Finding the max. weight I.S. Set **is ∈ P** [Golumbic04]

MCS Identification

Issue #2: efficiet MCS identification

Cast to **Minimum Flow problem:**



red = req.
green = flow

- Solve via inverse Ford-Fulkerson

Flow = W(MaxWIS) = **W(MaxCS)**
S/T cut = MaxWIS = **MaxCS**

Simple and fast:

- Based on the temporal graph
- No need to check usage at specific time instants
- **Incremental:** min flow used to prime the next serch node

MCS Identification

Issue #2: efficiet MCS identification

the **MCSs are Independent Sets** on the augmented graph => in exponential number

=> **In principle**, **NP-complete** identification

In practice, the graph has a transitive orientation

=> Finding the max. weight I.S. Set **is ∈ P** [Golumbic04]

Additionally:

We managed to apply some filtering algorithms for resource csts.

- Insufficient to check consistency on their own
- Sufficient to remove part of the probably infeasible values

MCS Identification

Issue #2: efficiet MCS identification

the **MCSs are Independent Sets** on the augmented graph => in exponential number

=> **In principle**, **NP-complete** identification

In practice, the graph has a transitive orientation

=> Finding the max. weight I.S. Set **is ∈ P** [Golumbic04]

Still an issue:

The MCS is not minimal and may be bad for branching

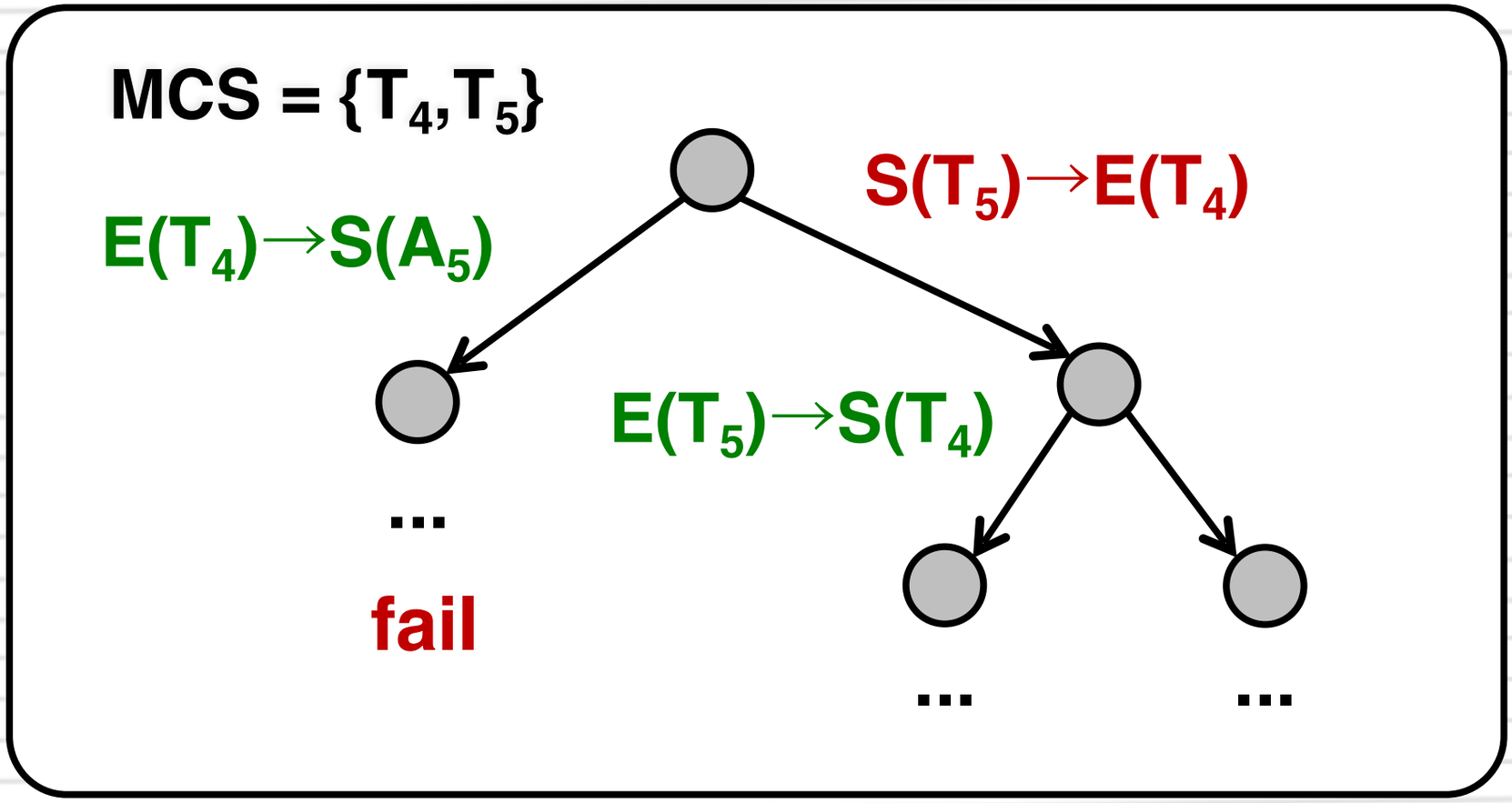
- **Greedy minization:** iteratively remove activities according to a heuristic quality estimate
- Chosen heuristics: preserved space [Laborie '95]

Branching

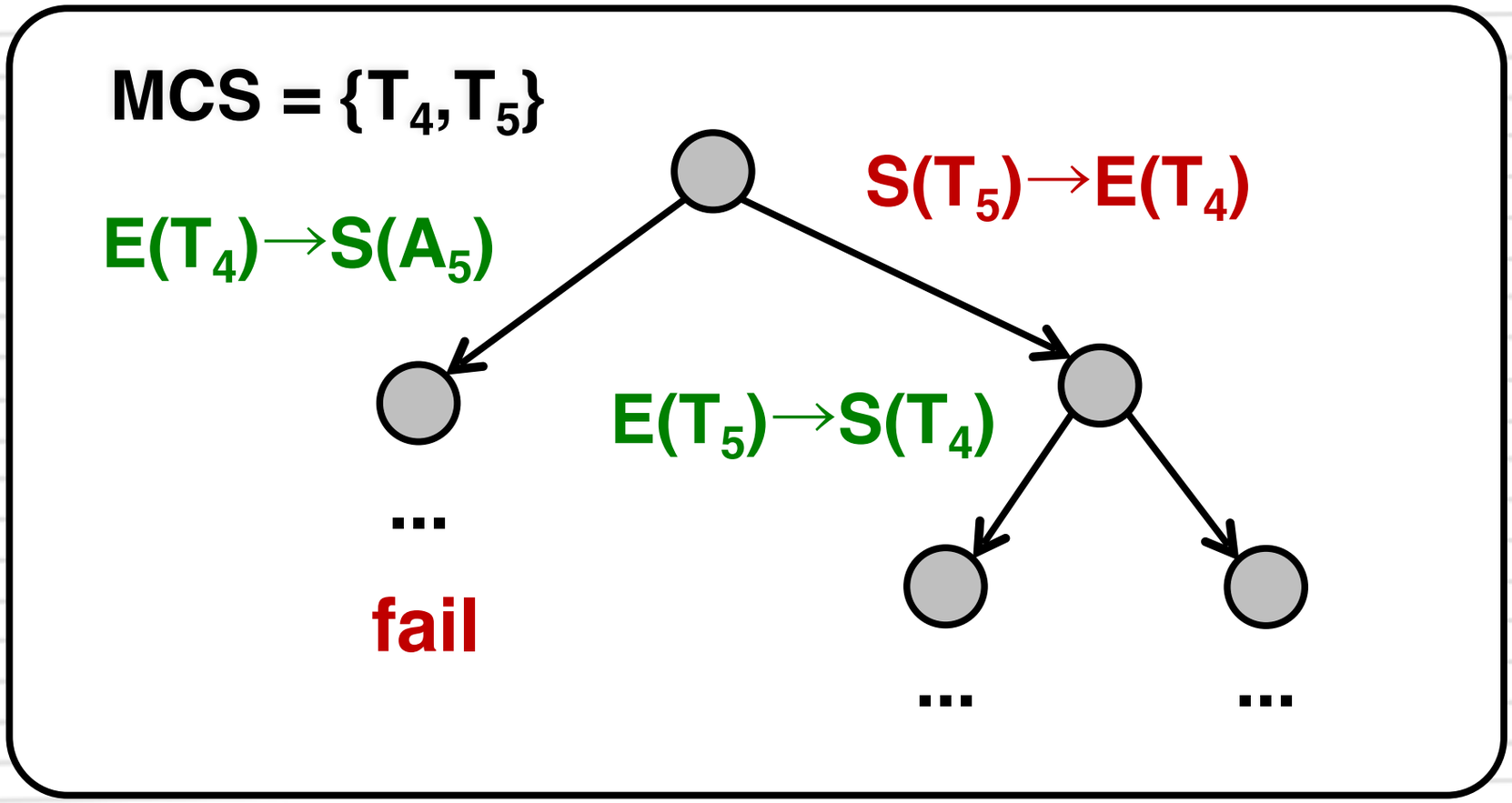
Given a **target MCS**, branch over **possible resolvers**

- Left branch: **post** a resolver
- Right branch: **forbid** a resolver

Search stops as soon as a resource **feasible** schedule is found (no optimization is performed at this stage)



Branching



- Optimization is performed by solving a **sequence of feasibility problems**
- A constraint on the chosen objective is iteratively tightened following a binary search scheme

Experimental Setting

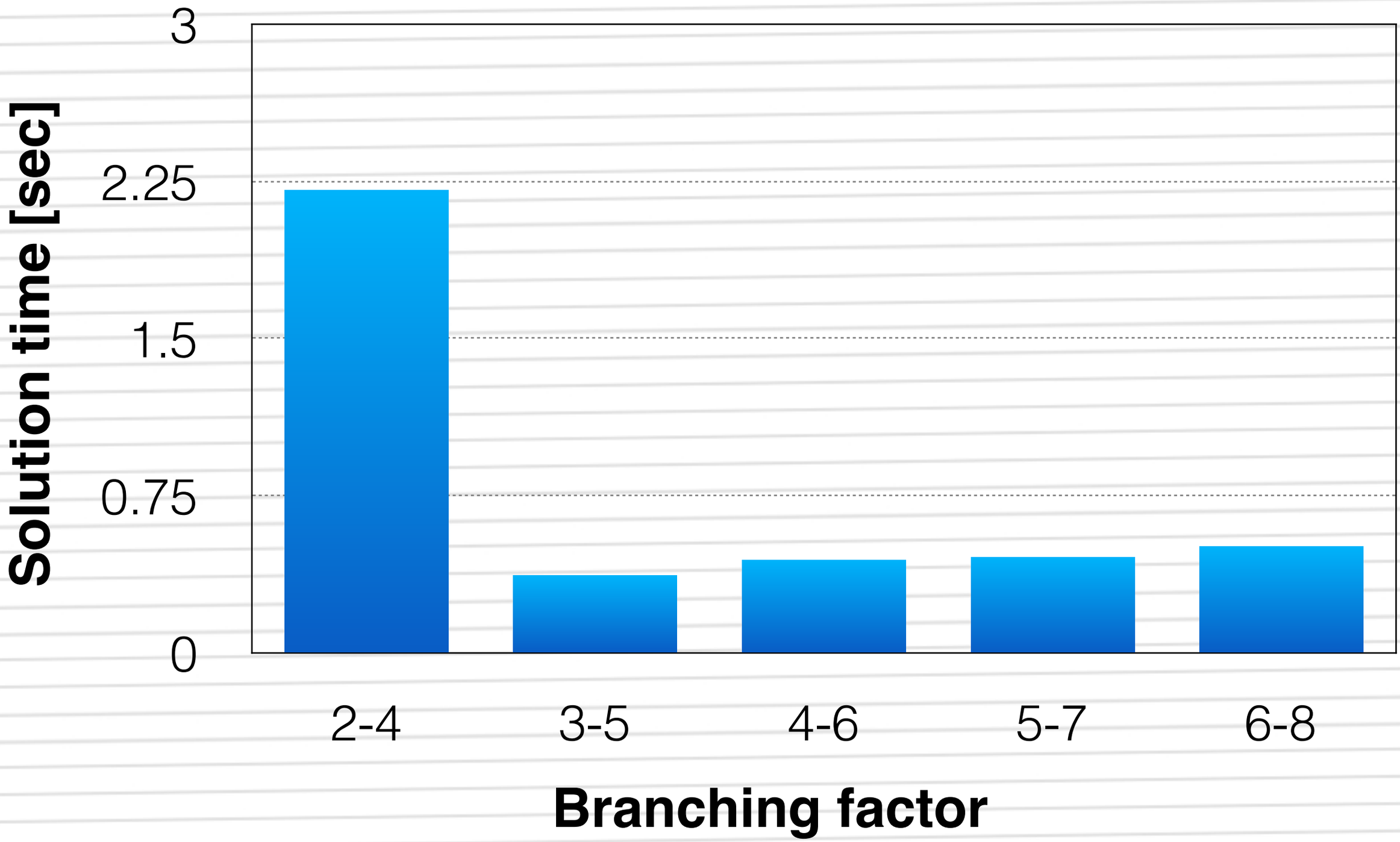
Implemeted on IBM-ILOG Solver 6.7, time limit 600s

Testing problem:

- For every activity $d_{min} \leq d_{max} \leq 2^* d_{min}$
- Platform:
  - 16 cores (unary, i.e. cap 1)
  - 2 communication ports per core
- Each **task** requires **1 unary resource**
- **Inter-task communications** require **10% to 90%** of the bandwidth **of an input port and of an output port**
- Instances with ≈ 80 activities, growing branching factor

Objective function: worst case execution time

Solution time, best WCET



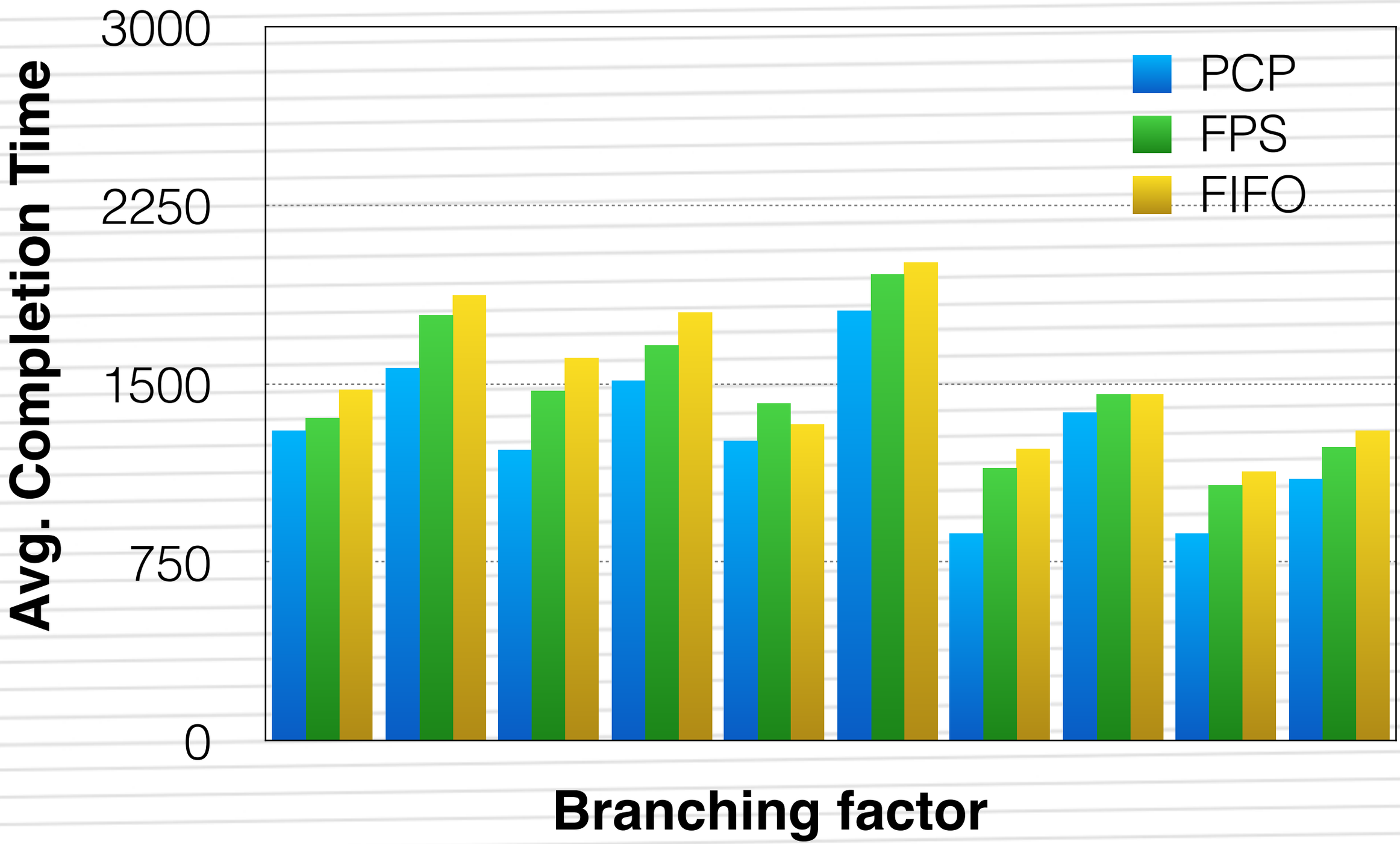
Average completion time minimization

Objective function: the completion time corresponding to the scenario where all tasks have average duration

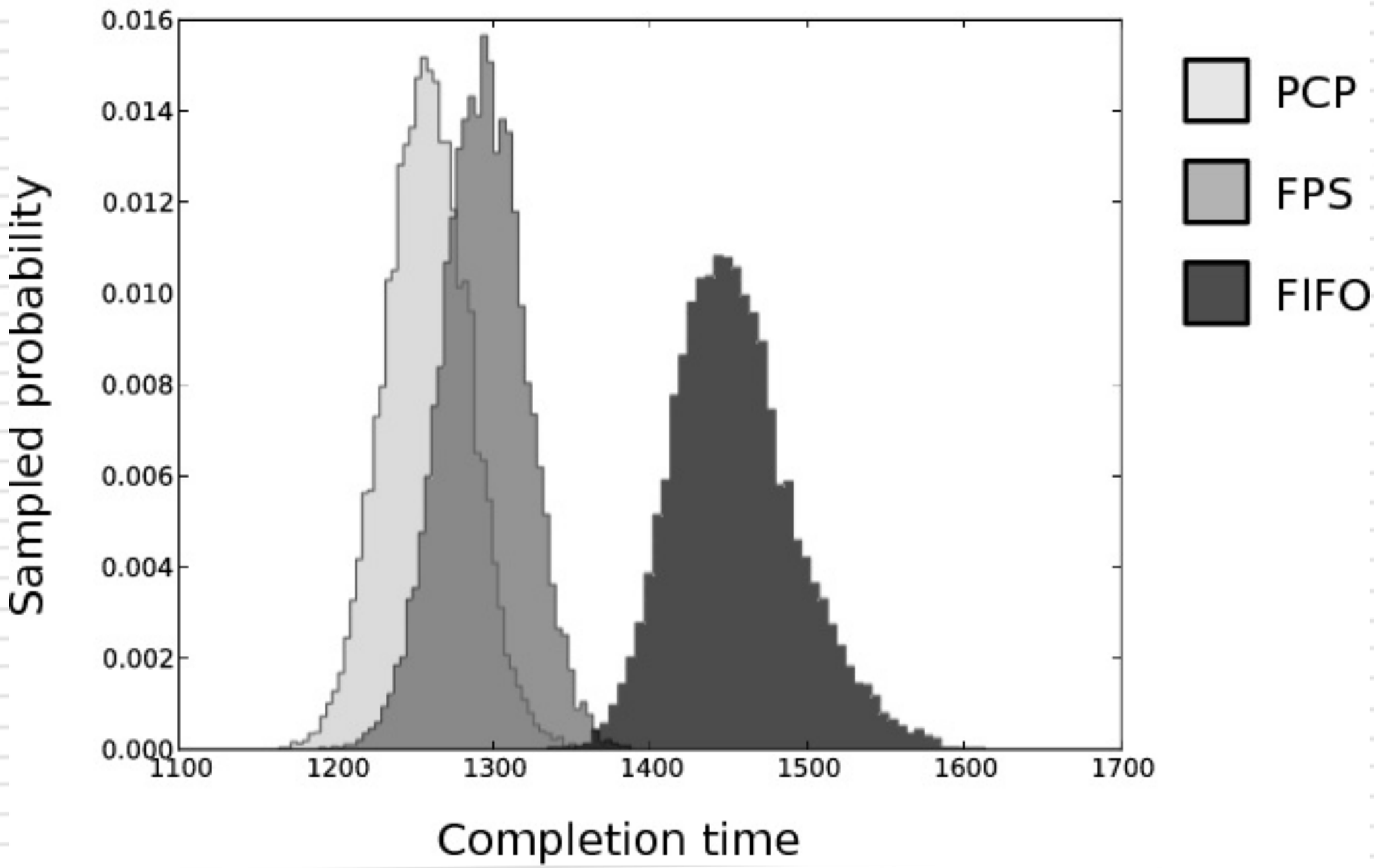
The **minimum application WCET** was used **as a global deadline**

- **Goal:** assess the quality of the PCP schedule in terms of average completion time
- Our PCP approach was **compared against:**
  - **Fixed Priority Scheduling**, optimized via Tabu Search
  - On-line **FIFO scheduling**
- Experiments on 10 instances from the previous benchmark
- **Evaluation via Monte-Carlo simulation**

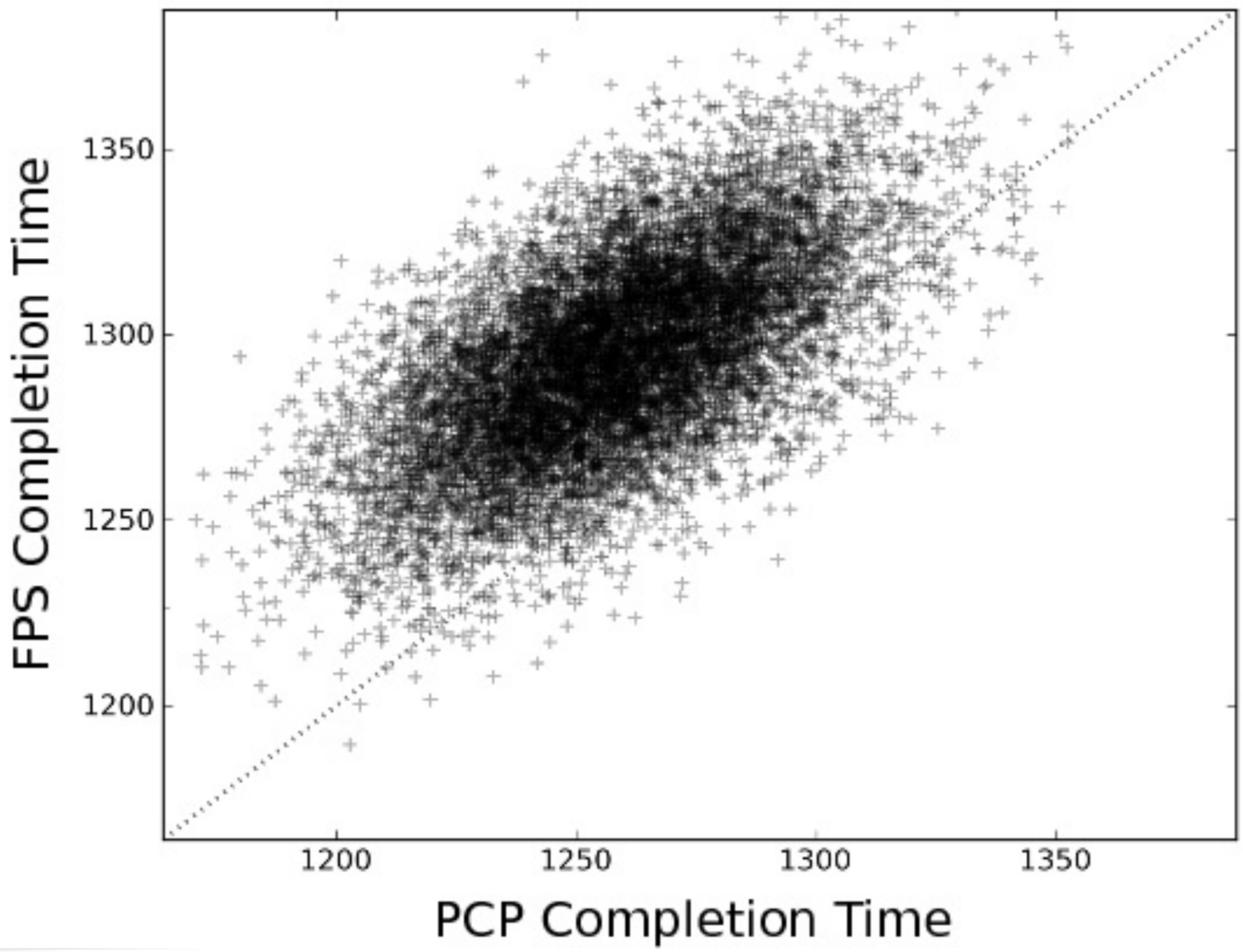
Average completion time minimization



Average completion time minimization



Average completion time minimization



Robust scheduling of task graphs under execution time uncertainty

Michele Lombardi

Michela Milano

Luca Benini

More advanced on-line schedulers

The on-line scheduler must:

- Start a task when the predecessors are over
- Start a task within the restricted time window

The on-line scheduler can be smarter than this:

- The scheduler <u>can decide how much to delay a task</u> (or not to delay it at all)
- The scheduler <u>can resolve resource conflicts on-line</u>

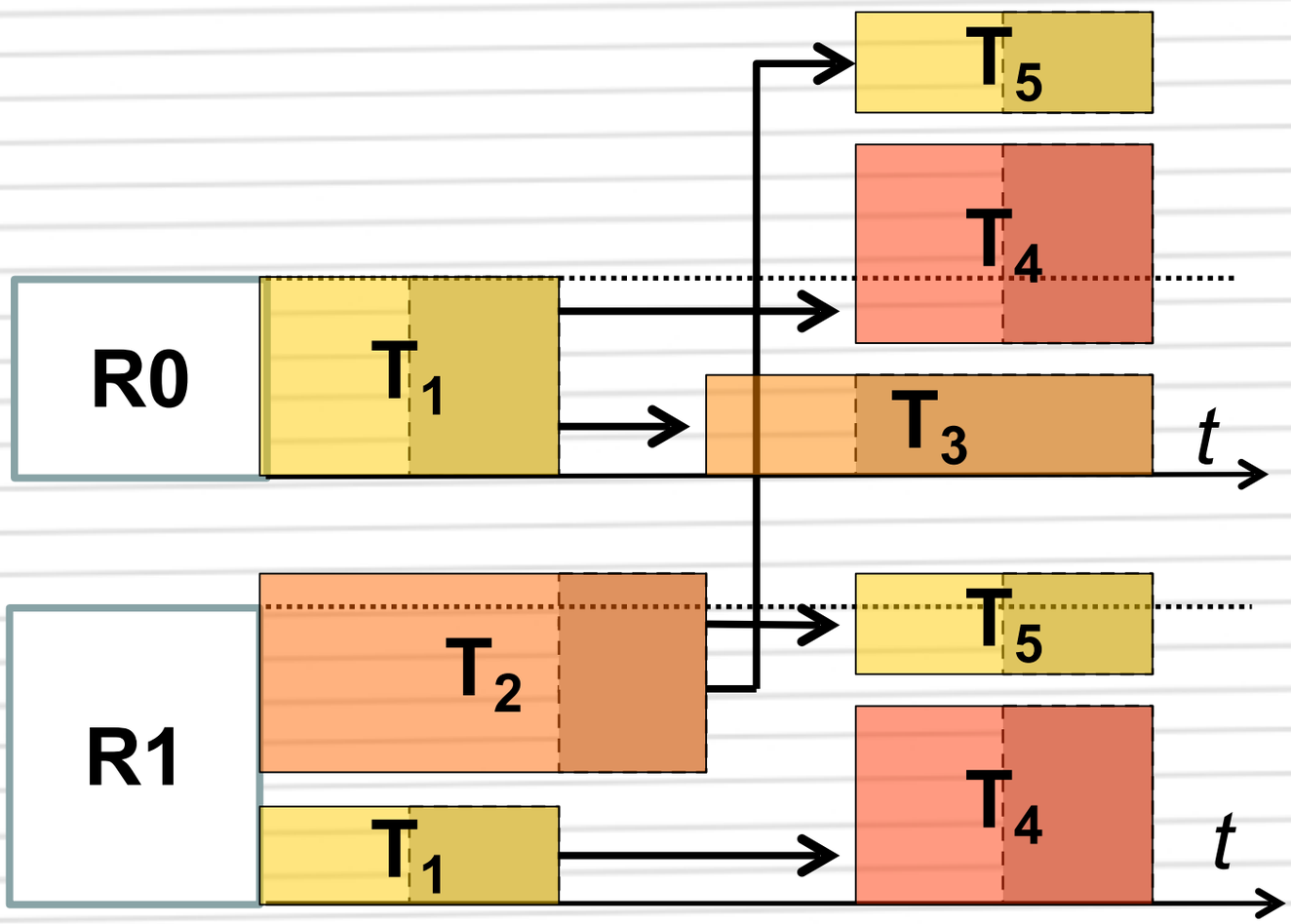
One more ingredient: Self Timed Execution (STE)

"activities are started by the dispatcher as soon as they are ready"

- Reduces the number of possibly occurring conflicts
- Introduces some complications in the solution algorithm

STE: Resource Conflicts

Some MCS should not be taken into account



Conflict Sets (no S.T.E.):

$\{A_1,A_5\}$, $\{A_3,A_4,A_5\}$, $\{A_4,A_5\}$, $\{A_4,A_5\}$

Conflict Sets (with S.T.E.):

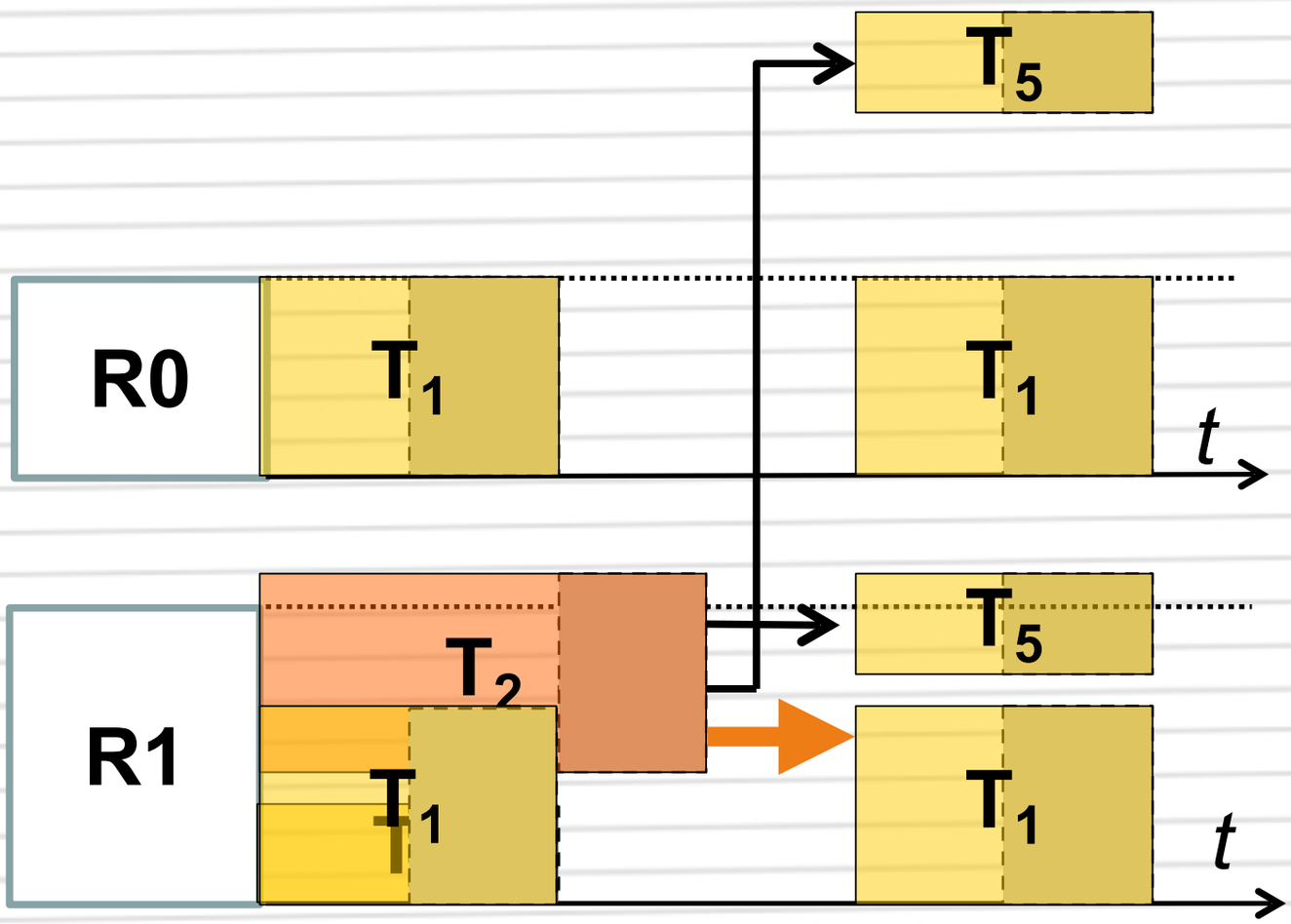
$\{A_3,A_4,A_5\}$, $\{A_4,A_5\}$, $\{A_4,A_5\}$

Conflict $\{A_1,A_5\}$ never occurs

BUT:

STE: Resource Conflicts

Some MCS should not be taken into account



Conflict Sets (no S.T.E.):

$\{A_1,A_5\}$, $\{A_3,A_4,A_5\}$, $\{A_4,A_5\}$, $\{A_4,A_5\}$

Conflict Sets (with S.T.E.):

$\{A_3,A_4,A_5\}$, $\{A_4,A_5\}$, $\{A_4,A_5\}$

Conflict $\{A_1,A_5\}$ never occurs

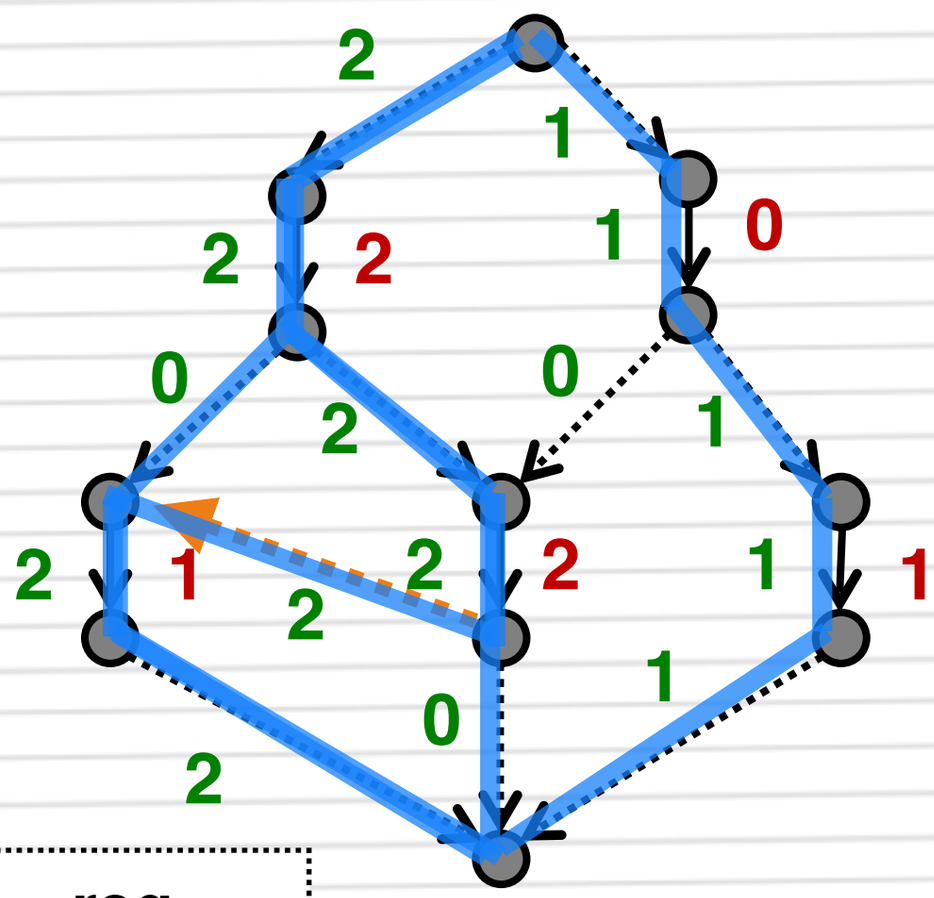
BUT: Suppose:

1. A1 uses 2 units of R1
2. $A_2 \rightarrow A_1$

new conflicts may arise due to search decisions

STE: Resource Conflicts

The graph for the min flow problem is **augmented with the <u>implied</u> precedence relations**, due to the self timed execution policy

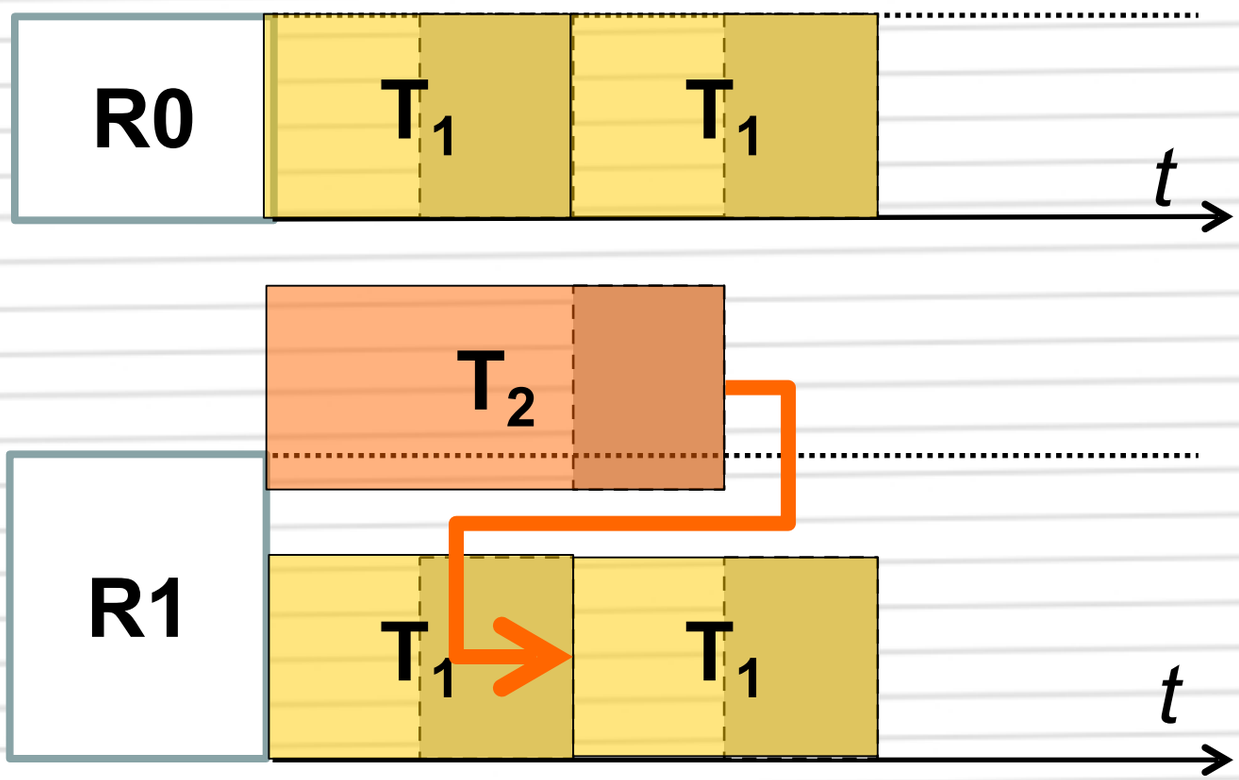


red = req.
green = flow

- **The flow value** (maximum weight of a Conflict Set) **may reduce**
- **Implied arcs may be removed** due to search decisions
- At each search node, the flow value is repaired to accommodate for removed arcs

STE: Temporal Infeasibilities

A1 uses 2 units of R1



- Suppose A_2 is scheduled at 0
- Propagation shifts A_1

A schedule with time gaps cannot be executed:

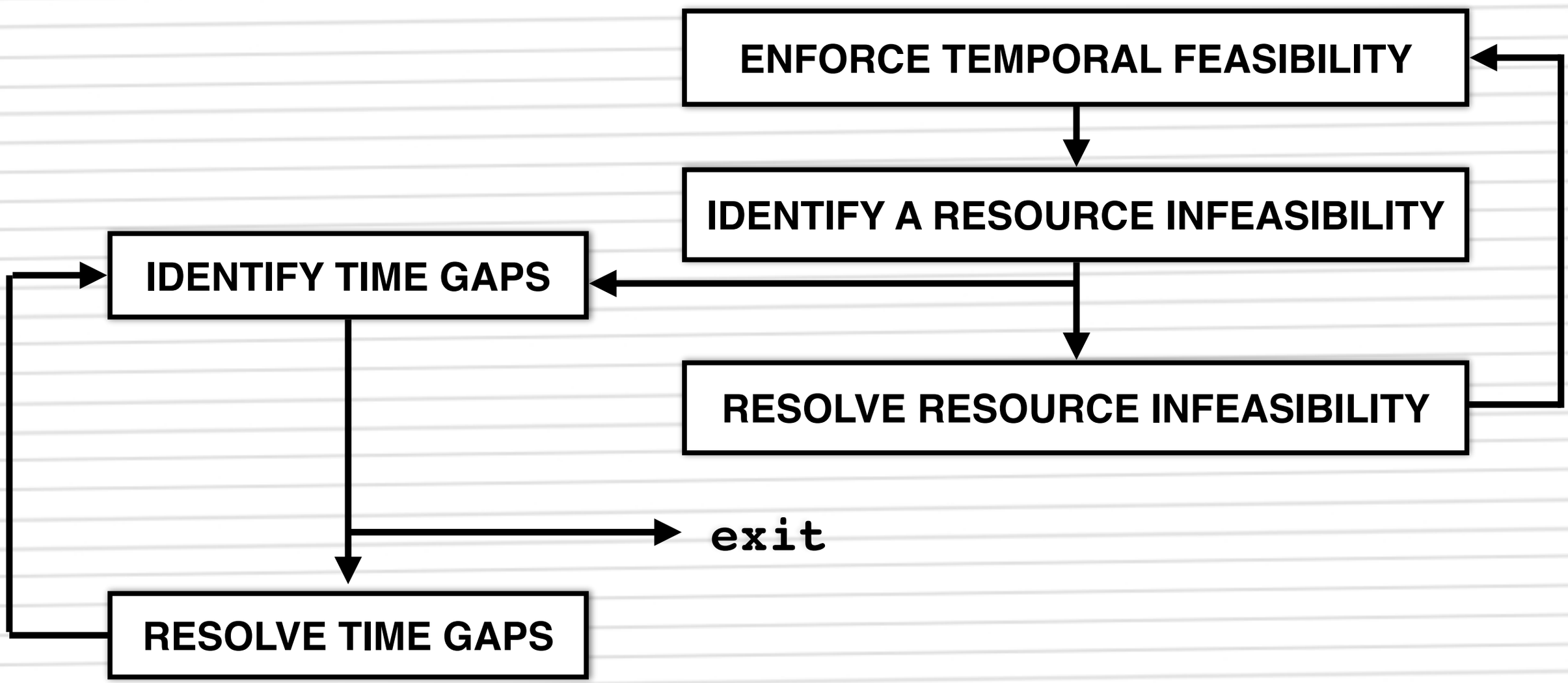
$$start(T_j) = \max_{T_i \prec T_j} end(T_i)$$

- The schedule can be fixed by adding a new precedence constraint

Issue 3: **stopping search** requires no conflicts, and **no time gaps**

Scheme of the solution approach

- A partial schedule can now be time infeasible
- Time gaps are resolved by adding precedence constraints, similarly to MCSs



Robust scheduling of task graphs under execution time uncertainty

Michele Lombardi

Michela Milano

Luca Benini