Mitra Nasri Nasrabadi
m.nasri@tue.nl
Assistant professor (tenure track)
Electronic Systems Group
Department of Electrical Engineering

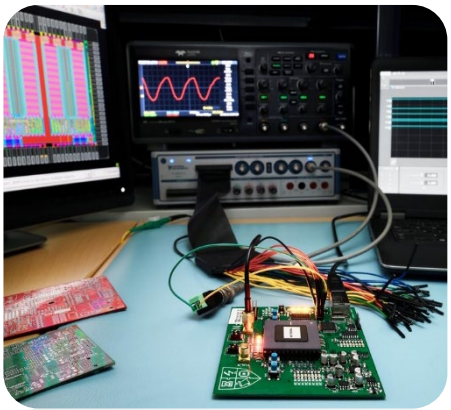
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Research Roadmap

Dependable Real-Time Cyber-Physical Systems

Department of Electrical Engineering

Electronic Systems (ES) Group



My background

Ph.D. 2009 to 2015
- The University of Tehran
- (in collaboration with TU-Kaiserslautern)

Won a DAAD scholarship in 2013

Postdoc 2015 to 2016
- TU-Kaiserslautern

Won an Alexander von Humboldt Fellowship in 2016

Postdoc fellow 2016 to 2018
- Max Planck Institute for Software Systems

Won Delft Technology Fellowship Award in 2018

Assistant professor (tenure track) October 2018 to August 2020
- TUDelft [Embedded and Networked Systems Group, EEMCS]

Assistant professor (tenure track) February 2020
- TU/e [Electronics Systems Group, Department of Electrical Engineering]

Research area:

Real-time cyber-physical systems (CPS)

Research area:

Real-time cyber-physical systems (CPS)



Safety

- Human life
- Environment

Time-predictability

- A late (or missed) actuation may cause safety violation
- Example: breaking, air-bag inflation, etc.

Correct response

Functional correctness

+

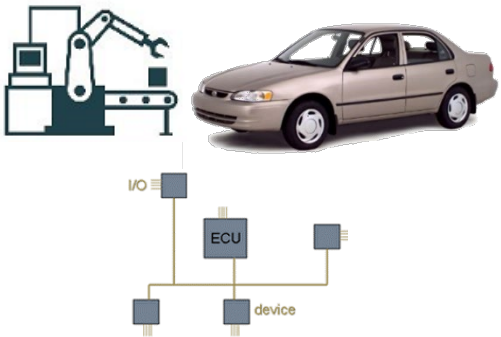
Temporal correctness

On time response

Fast ≠ predictable

Why guaranteeing temporal correctness is hard?

Back then

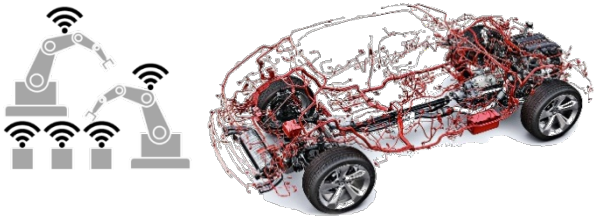


A few computing nodes and control loops

Simple hardware and software **architecture**

Simple, predictable, and easier-to-analyze

Now (and future)



Complex software (application workloads) running on heterogeneous computing environment

Intensive I/O accesses

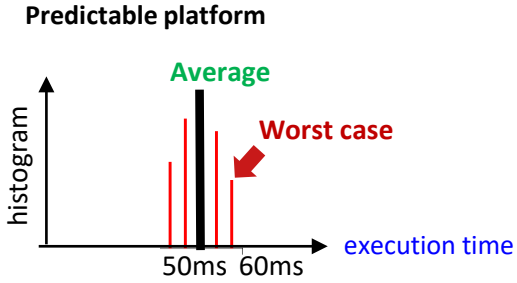
Use of hardware accelerators (GPUs, FPGA, co-processors, etc.)

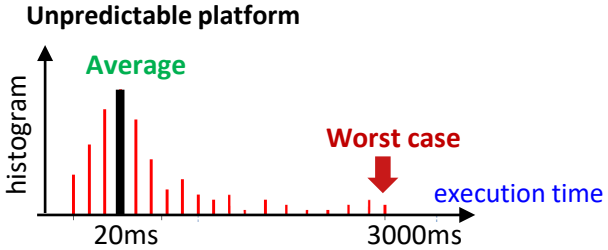
Computation offloading (to the cloud, edge, etc.)

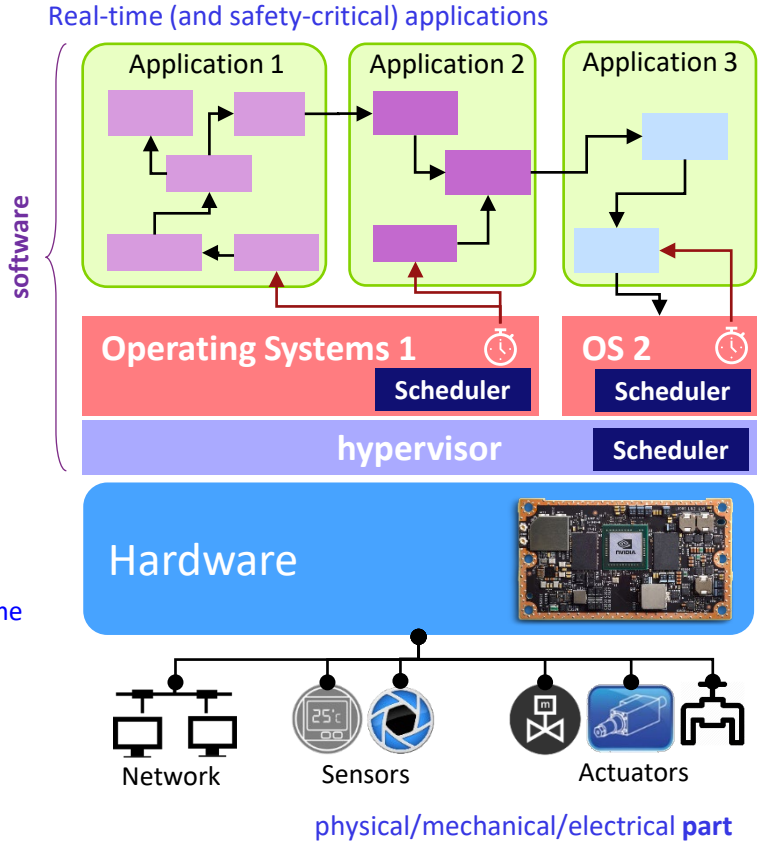
Heterogeneous communication

Complex, less predictable, and harder-to-analyze

Why guaranteeing temporal correctness is hard?







My main research questions

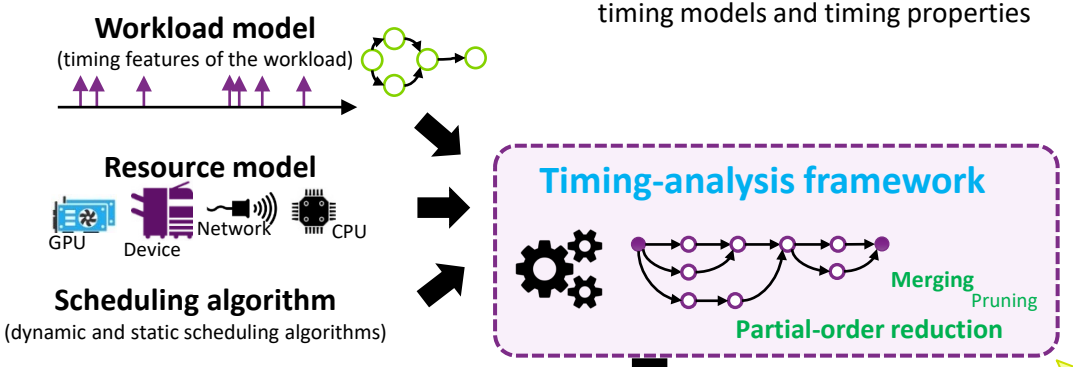
How to **analyze** the timing behavior of a given system?

Why?

To verify temporal correctness in an accurate and scalable way

How?

- By designing a formal verification engine that is **dedicated** to timing models and timing properties



Worst-case performance bounds

(for para-functional properties)

- **Lower and upper bounds on**
  - the end-to-end response time
  - latency, throughput, jitter, quality of service/control, etc.
- **Counter examples** in case of timing violation

3000x faster than generic timing verification tools such as **UPPAAL** **while being highly accurate**

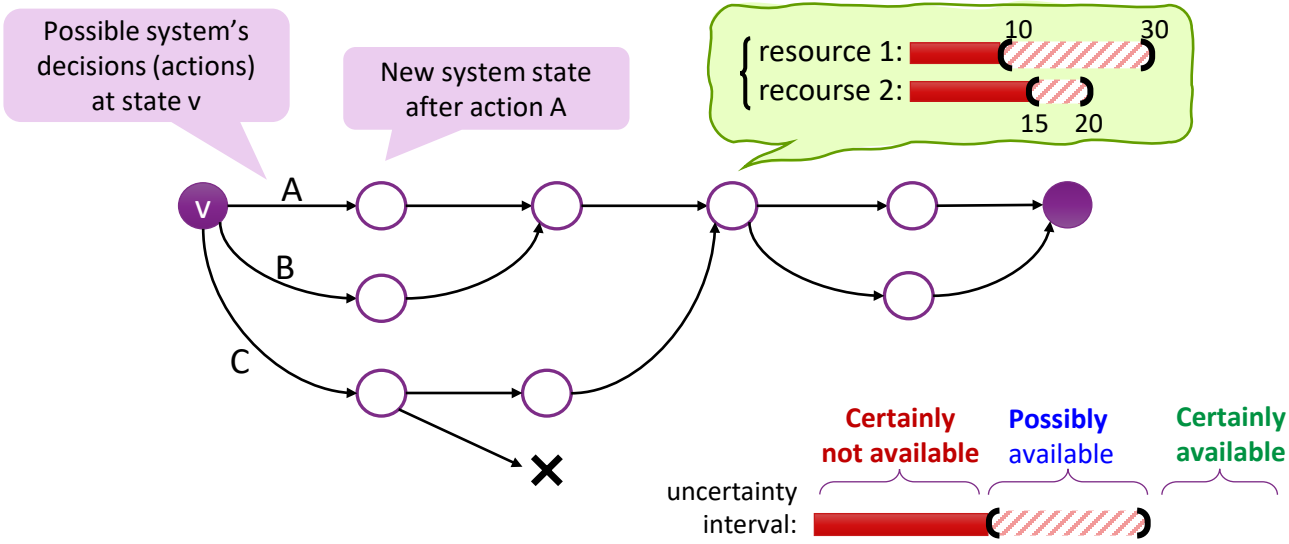
My main research questions

Timing-analysis framework
(it is a formal verification engine dedicated to **timing models** and **timing properties)**

How to **analyze** the timing behavior of a given system?

Together with my team, we have designed a **reachability analysis** that

- Uses **uncertainty intervals** to combine uncertainties in the platform and task activation patterns
- **Merges states whose future is similar**
- **Does not** explore paths that **do not contribute to the worst-case behavior**



My main research questions

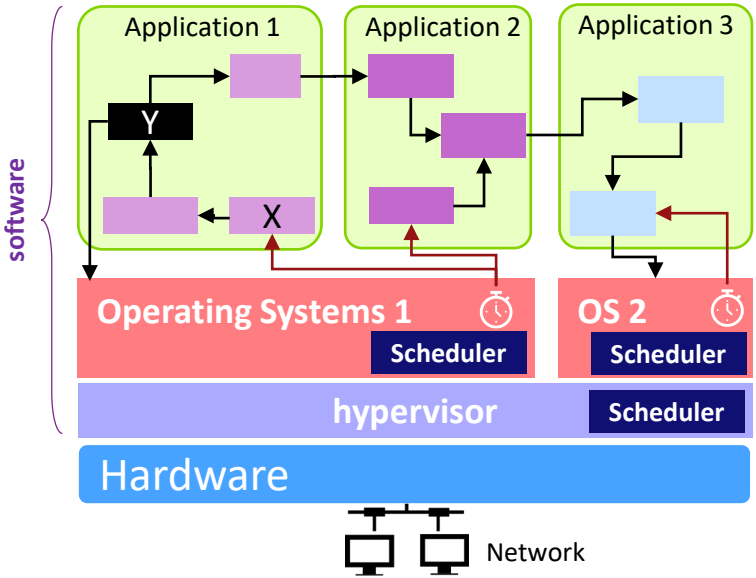
How to derive the **timing model** of a complex system under partial observations?

Why?

Security and safety monitoring

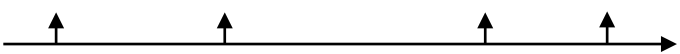
How?

- Regression-based machine learning
- Pattern recognition in time series
- Forecasting future events in time series using recurrent neural networks



Observable behavior
(Time intervals during which task Y accesses a resource)

Goal: verify if the activation pattern of task Y conforms its specification



My main research questions

How to **analyze** the **worst-case performance** of a **decision-making system** that uses **AI**?

How to design a **dependable** (trustable) **decision-making system** that uses **AI**?

How?

Timing-analysis framework
(it is a formal verification engine dedicated to **timing models** and **timing properties)**

Extend to

Analyze the **worst-case timing behavior** of a system scheduled by a **smart scheduler**

Are we ready to **trust** (and **depend on**) AI for **safety-critical systems**?

Why?

Other research interests

Secure and dependable real-time systems

Since 2019

How to **protect** systems against **security attacks** whose success depend on a certain timing criteria?

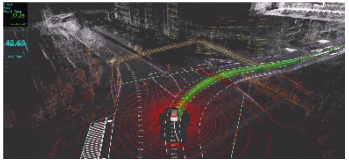
Example: the attack must happen immediately after the sampling

ROS Robot Operating System

Time-predictable ROS

Since 2020

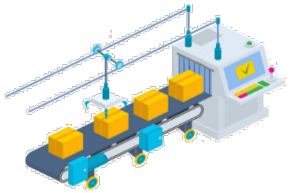
How to make **ROS2** more **time-predictable**?



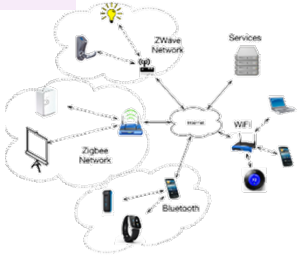
Efficient scheduling solutions with predictable performance

Since 2009

Scheduling in embedded systems



Product-line scheduling



Scheduling communications in IoT

Canon industrial printing

My current project involvements

How to **analyze** the timing behavior of a given system?

How to **design** and **configure** a system to have better timing predictability?

How to **manage resources** at **runtime**, and **adapt** to **dynamic changes** such that the system **continuously delivers correct and timely** service?

How to **analyze** the **worst-case performance** of a **decision-making system** that uses **AI**?

How to design a **dependable** (trustable) **decision-making system** that uses **AI**?

TRANSACT

H2020 project (1.4M€, 3 PhD students)
June 2021 to 2024

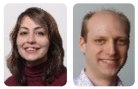
SAM-FMS

NWO project (600K€, 2 PhD students)
Sep. 2020 to 2024

My startup package

2 PhD positions

Working on a proposal



Would like to collaborate?
Please contact me

Questions?

Interested in collaboration?

- I can help you **improve the timing predictability** of your cyber-physical system or to see if it conforms your timing specifications.
- I need help from experts on *AI-based scheduling*, *formal verification and parity games*, and *optimization and resource allocation*

How to derive the **timing model** of a complex system under partial observations?

How to **analyze** the timing behavior of a given system?

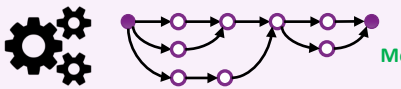
How to **design** and **configure** a system to have better timing predictability?

How to **manage resources** at **runtime**, and **adapt** to **dynamic changes** such that the system **continuously delivers correct and timely** service?

How to **analyze** the **worst-case performance** of a **decision-making system** that uses **AI**?

How to design a **dependable** (trustable) **decision-making system** that uses **AI**?

Timing-analysis framework
(it is a formal verification engine dedicated to **timing models** and **timing properties**)



Partial-order reduction

Mitra Nasri Nasrabadi
m.nasri@tue.nl
Assistant professor
Electronic Systems Group
Department of Electrical Engineering

State of the art

Closed-form analyses
(e.g., problem-window analysis)

- Fast
- Pessimistic
- Hard to extend

Exact analyses in generic formal verification tools (e.g., UPPAAL)

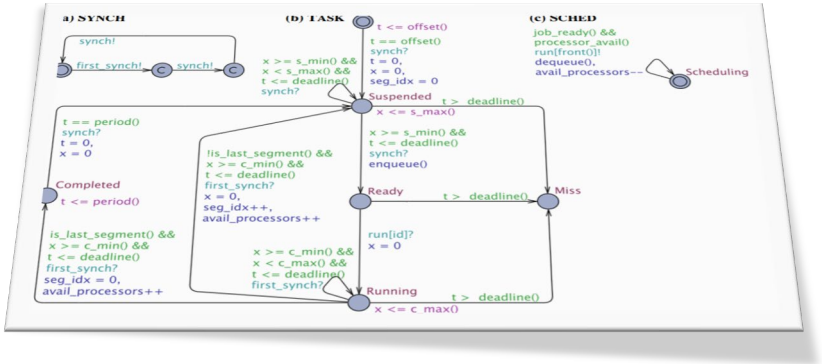
- Accurate
- Easy to extend
- Not scalable

$$R_i^{(0)} = C_i + \sum_{j=1}^{i-1} C_j$$

$$R_i^{(k)} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i^{(k-1)}}{T_j} \right\rceil C_j$$

...reemptive regions of lower-priority tasks. A response-time analysis of a DAG-based task-set with a limited-preemptive global fixed priority scheduler is computed by iterating the following equation until a fixed point is reached, starting with $R_k = len(G_k) + \frac{1}{m}(vol(G_k) - len(G_k))$:

$$R_k \leftarrow len(G_k) + \frac{1}{m}(vol(G_k) - len(G_k) + I_k^{hp} + I_k^{lp}) \quad (1$$



State of the art

Closed-form analyses
(e.g., problem-window analysis)

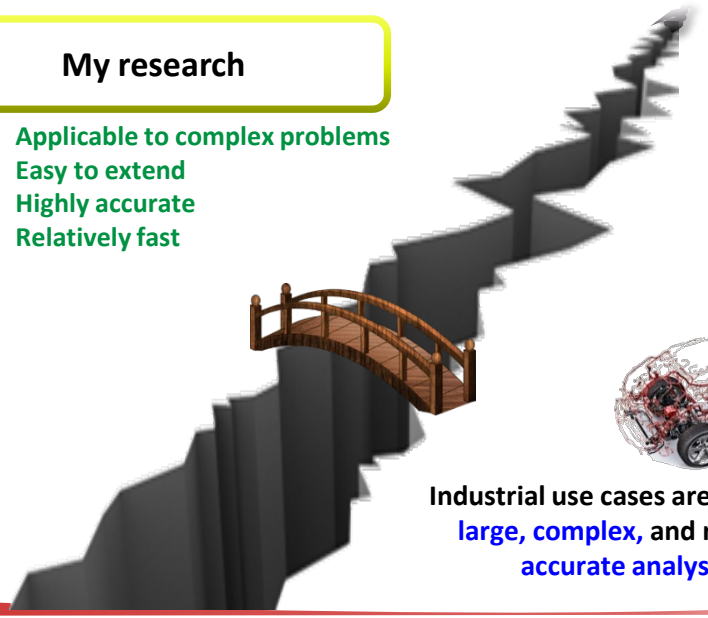
- Fast
- Pessimistic
- Hard to extend

Exact analyses in generic formal verification tools (e.g., UPPAAL)

- Accurate
- Easy to extend
- Not scalable

My research

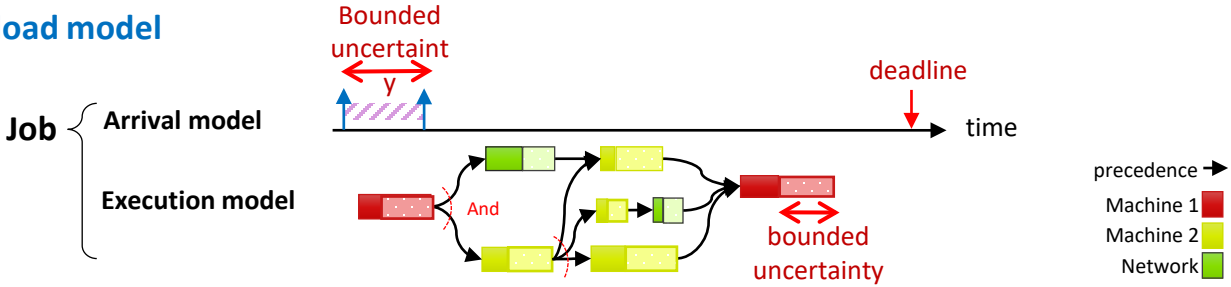
- Applicable to complex problems
- Easy to extend
- Highly accurate
- Relatively fast



Industrial use cases are typically large, complex, and require accurate analysis

The response-time analysis problem

Workload model



Resource model

How many resources of which type (and with what access costs) are available?

Scheduler model

How are the resources governed (scheduled)?

Response-time analysis problem

Given **a set of jobs, resources, and scheduling policies,**
Determine **the worst-case response time of each job**

Why the problem is hard?

The preemptive version of this problem (which is believed to be easier than the non-preemptive one), is already NP-Complete

One of the simplest forms of the problem:

Response-time analysis problem

Given

- **a set of non-preemptible jobs** (with a given arrival interval, execution time, and deadline)
- **scheduled by a fixed-priority scheduling policy**
- **on a uniprocessor platform,**

Determine

the worst-case response time of each job

Why the problem is hard?

One of the simplest forms of the problem:

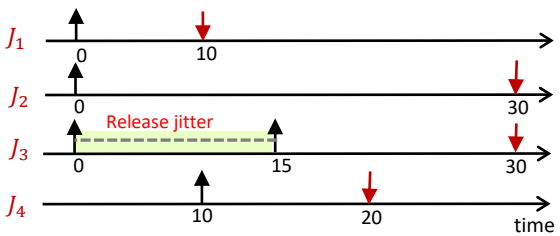
Response-time analysis problem

Given

- **a set of non-preemptible jobs** (with a given arrival interval, execution time, and deadline)
- **scheduled by a fixed-priority scheduling policy**
- **on a uniprocessor platform,**

Determine

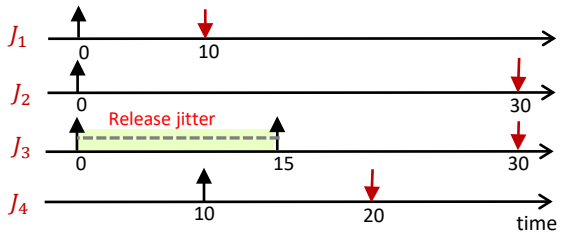
the worst-case response time of each job



Job	Release time Min	Release time Max	Deadline	Execution time Min	Execution time Max	Priority
J_1	0	0	10	1	2	high
J_2	0	0	30	7	8	medium
J_3	0	15	30	3	13	low
J_4	10	10	20	1	2	high

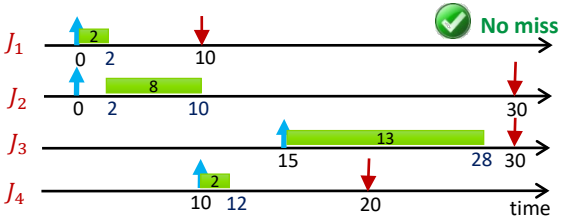
Earliest release time | Latest release time | BCET | WCET

Why the problem is hard?

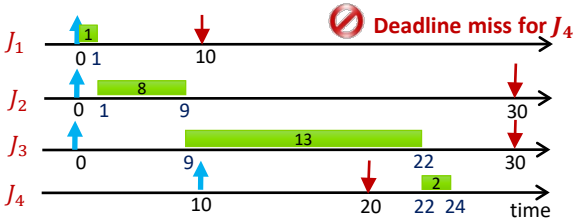


Job	Release time Min	Release time Max	Deadline	Execution time Min	Execution time Max	Priority
J_1	0	0	10	1	2	high
J_2	0	0	30	7	8	medium
J_3	0	15	30	3	13	low
J_4	10	10	20	1	2	high

Execution scenario 1: jobs are released very late and have their largest execution time.



Execution scenario 2: jobs are released very early and have their largest execution time except for J_1.



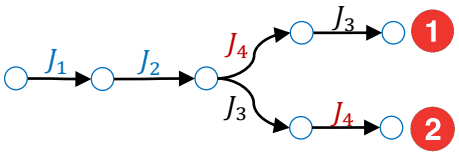
How should we find such a worst-case scenario?

Why the problem is hard?

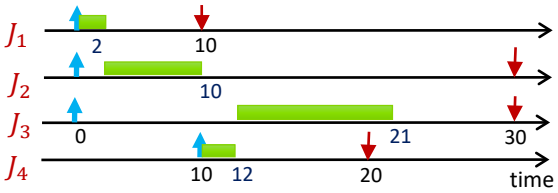
Naively enumerating all possible combinations of release times and execution times (a.k.a. execution scenarios) **is not practical**

Observation:

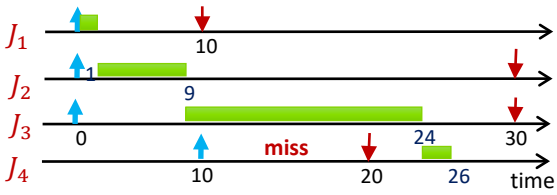
There are fewer permissible job orderings than schedules



Example for path 1



Example for path 2



- **2 possible job ordering**
- **1200 different combinations for release times and execution times**

Why the problem is hard?

Naively enumerating all possible combinations of release times and execution times (a.k.a. execution scenarios) **is not practical**

Observation:

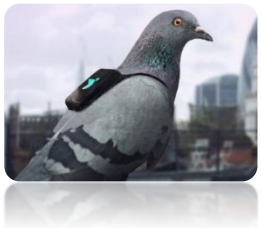
There are fewer permissible job orderings than schedules

Solution idea:

We use job-ordering abstraction to build a graph that abstracts all possible schedules

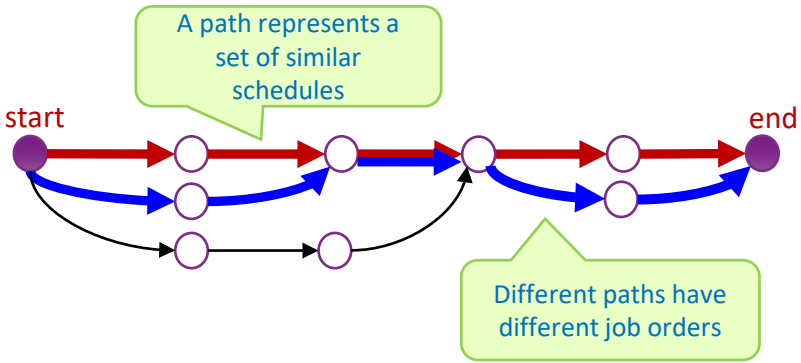
It is called the "schedule-abstraction graph"

Goal: an *accurate* and *efficient* analysis



Response-time analysis using schedule-abstraction graphs

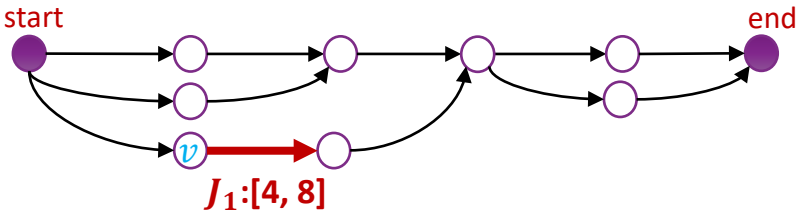
A path aggregates all schedules with the same job ordering



Response-time analysis using schedule-abstraction graphs

A **path** aggregates all schedules with the same job ordering

A **vertex** abstracts a system state and an **edge** represents a dispatched job

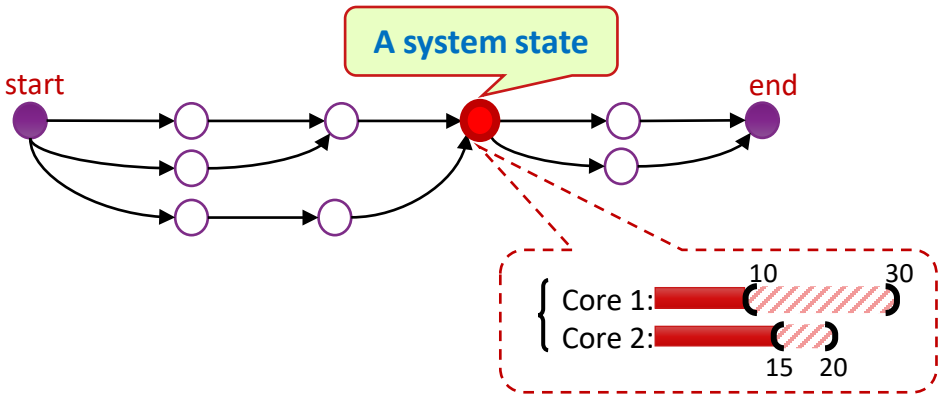


Response-time analysis using schedule-abstraction graphs

A path aggregates all schedules with the same job ordering

A vertex abstracts a system state and an edge represents a dispatched job

A state is labeled with the finish-time interval of any path reaching the state



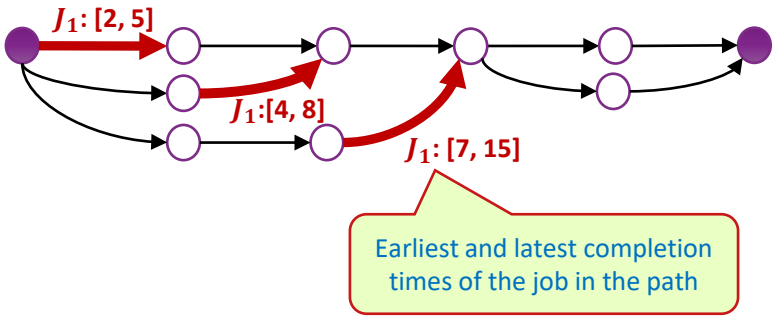
Response-time analysis using schedule-abstraction graphs

A path aggregates all schedules with the same job ordering

A vertex abstracts a system state and an edge represents a dispatched job

A state represents the finish-time interval of any path reaching that state

Obtaining the response time:

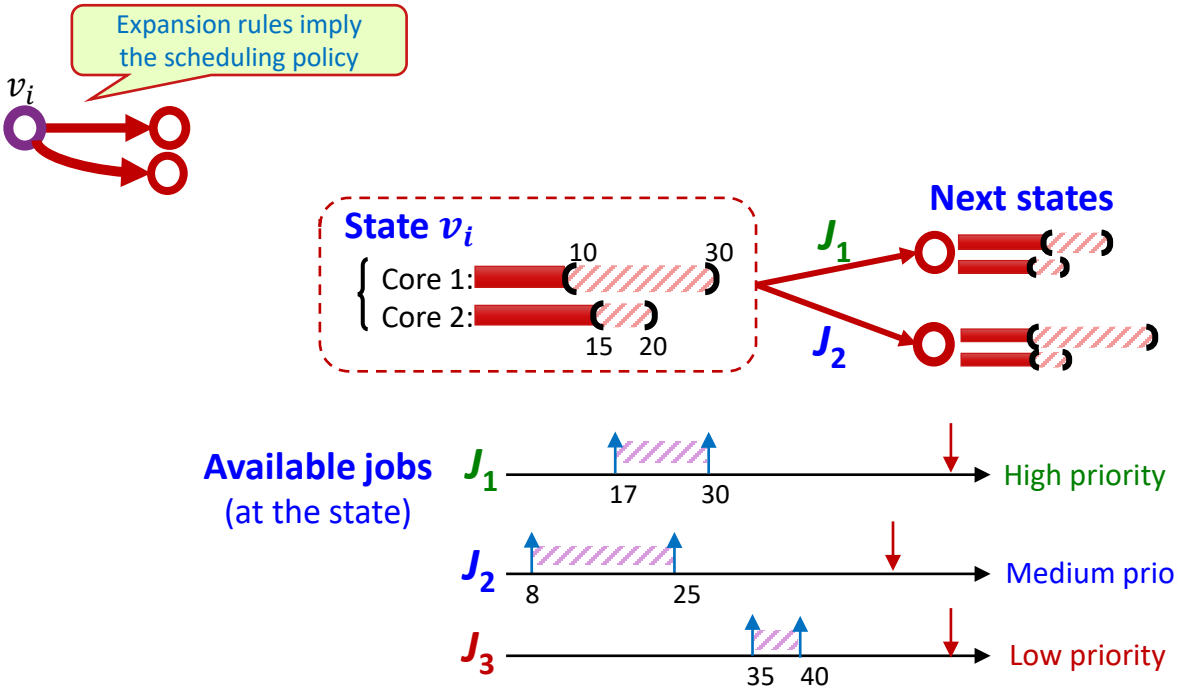


Best-case response time = **min** {completion times of the job} = **2**

Worst-case response time = **max** {completion times of the job} = **15**

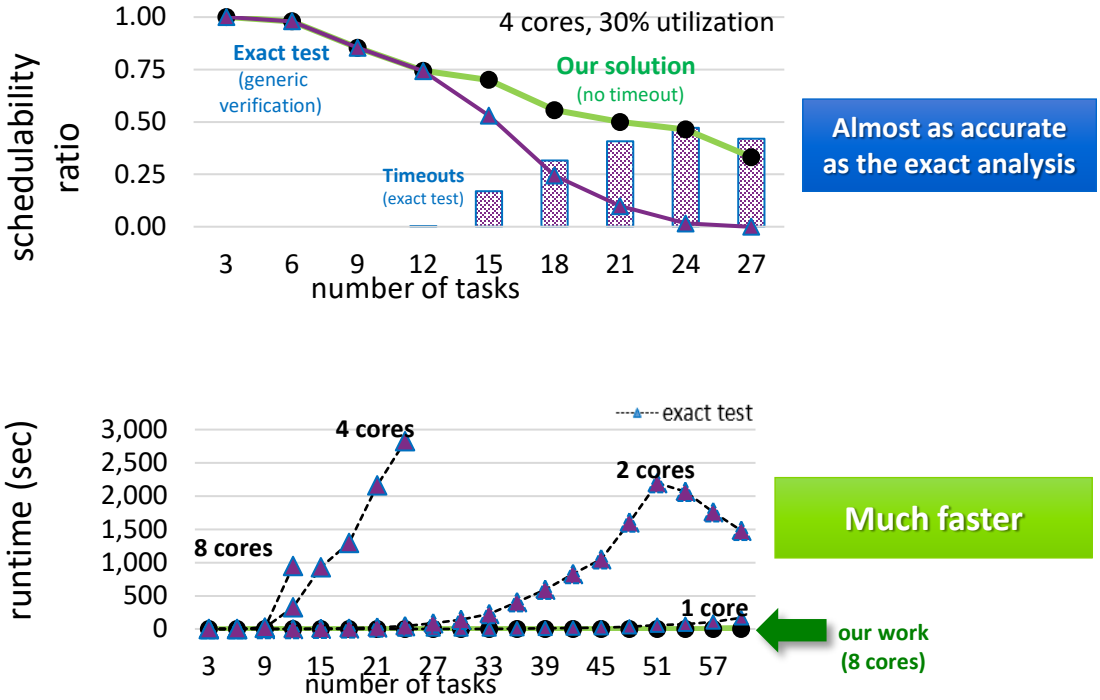
Building the schedule-abstraction graph

Expanding a vertex:
(reasoning on uncertainty intervals)



Taste of results

Comparison with an **exact schedulability test** implemented in UPPAAL, a **generic model-verification tool**



Schedulability = ratio of systems deemed schedulable to the total evaluated systems

My network

The Netherlands


Benny Akesson (TNO-ESI, UvA)


Claudia Hauff (TUD)


Jeroen Voeten (TU/e)


Geoffrey Nelissen (TU/e)


Twan Basten (TU/e)


Marc Geilen (TU/e)


Kees Goossens (TU/e)


Sander Stuijk (TU/e)

...

Europe and USA (Germany, Sweden, Italy, Portugal, Austria, and UK)


Björn Brandenburg (MPI)


Karl-Erik Arzen (Lund)


Anton Cervin (Lund)


Eduardo Tovar (CISTER)


David Broman (KTH)


Robert Davis (York)


Arne Hamann (Bosch)


Marko Bertogna (UniMore)


Gerhard Fohler (TU-KL)


Ramon Serena (TTTech)


Sanjoy Baruah (WashU)


Tam Chantem (VTech)


Ryan Gerdes (VTech)


Gedare Bloom (UCCS)

...