Support for Limited-Preemptive Fixed-Priority Scheduling

- an evolutionary step still facing research challenges -

Reinder J. Bril
associate professor at TU/e

SAN

TU/e Technische Universiteit Eindhoven University of Technology

Where innovation starts

NGOSCPS, Montreal, 2019 April 15th

Background and motivation

- **Fixed-priority preemptive scheduling (FPPS):**
  - **described in standards, e.g. POSIX®, OSEK/AUTOSAR;**
  - **documented in practitioners handbook by CMU/SEI;**
  - **supported by courses and analysis tools;**
  - **supported by most COTS RTOS;**
  - **de facto standard in industry.**

- **Disadvantage: *arbitrary* pre-emptions.**

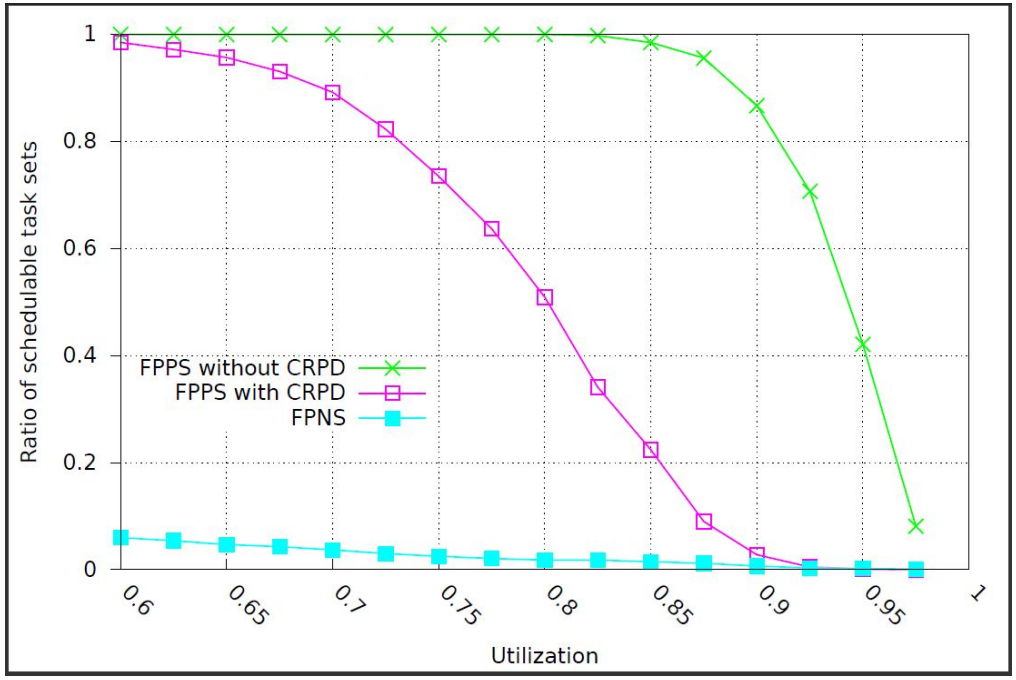
Background and motivation

- **Consequences of *arbitrary* preemptions:**
  - **inefficient memory use (e.g. stack);**
  - **inefficient CPU use**
    - **high run-time overheads (e.g. caches).**

- **Fixed-priority *non*-preemptive scheduling (FPNS) instead?**
  - **No, due to reduced schedulability.**

Background and motivation

- **Ratio of schedulable task sets [3].**



Experiment:
- n = 10 tasks;
- T_i (= D_i) $\in$ [10,1000] ms;
- U_i task utilization by Uunifast*;
- 1000 systems per U set util.

Cache related:
- N = 512 cache sets, BRT = 8µs;
- U^C = 4 total cache utilization (total # of ECBs is 512 x 4);
- U_i^C task cache util. by UUnifast;
- 40% of ECBs are UCBs.

CRPD has a *significant* impact on schedulability!

*E. Bini and G. Buttazzo, RTSJ, 2005.

[3] S. Altmeyer, R.I. Davis and C. Maiza, RTSJ, 2012.

Background and motivation

- **Observations:**
  - **DMPO is no longer optimal for "FPPS with CRPD" for constrained deadlines;**
  - **"FPPS with CRPD" is not compatible with Audsley's Optimal Priority Assignment (OPA) algorithm.**
- **Open problem:**
  - **Computational tractable method to determine optimal priorities for "FPPS with CRPD".**

[32] H.-N. Tran et al., ETFA, 2015.

Background and motivation

- **Alternative: *Limited pre-emptive* FPS (LP-FPS)**
  - ***Advantages*:**
    - ***Reduced* memory requirements (FPPS);**
    - ***Reduced* cost of arbitrary pre-emptions (FPPS);**
    - ***Improved* schedulability of task sets (FPPS & FPNS).**
  - ***Disadvantage*:**
    - **Added complexity, both analysis and mechanisms.**

Background and motivation

- ***Cost-effectiveness improvements*** **brought by LP-FPS**
  - i. **Application $\mathcal{A}$ may now run on platform $\mathcal{P}$;**
  - ii. **Next to $\mathcal{A}$ we may run $\mathcal{A}'$ on $\mathcal{P}$;**
  - iii. **$\mathcal{A}$ may now run on platform $\mathcal{P}'$, with $\mathcal{P}' < \mathcal{P}$.**

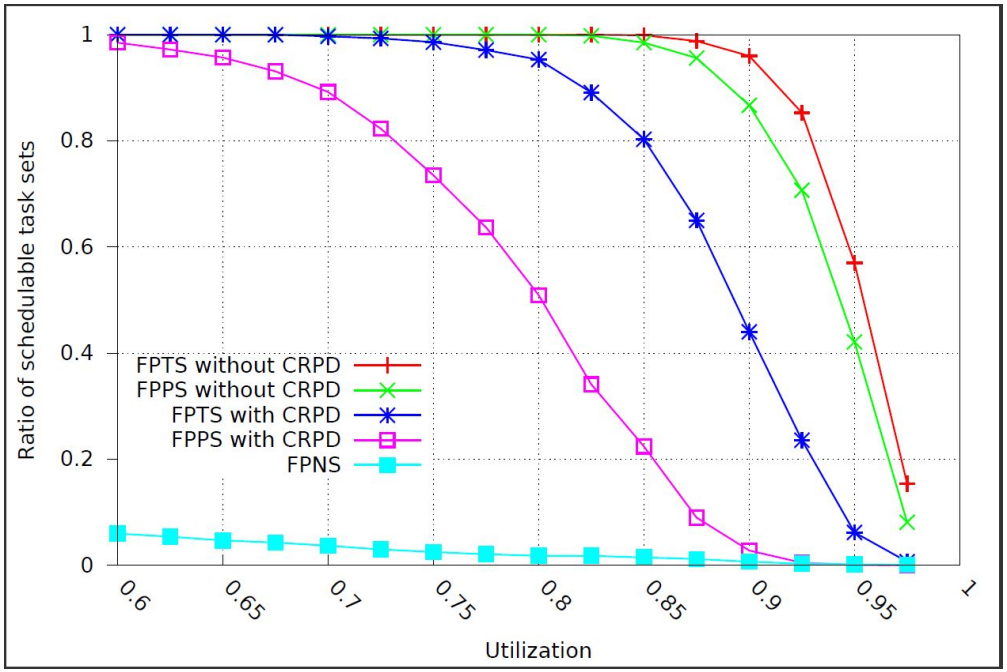
- **LP-FPS is an *evolutionary* step**
  - **…but only limited support by COTS RTOS;**
  - **…and still facing research challenges.**

Basic LP-FPS schemes

1. **Pre-emption thresholds (FPTS)**
   - **Scheduling algorithm:**
     - **fixed pre-emption threshold (θ)**
       - $\forall_i \theta_i = \pi_i \Rightarrow$ **FPPS**
       - $\forall_i \theta_i = \pi_1 \Rightarrow$ **FPNS**
   - **FPTS generalizes both FPPS and FPNS.**

FPTS, FPPS, FPNS, and CRPD

- **Ratio of schedulable task sets [11].**



Experiment:
- n = 10 tasks;
- T_i (= D_i) $\in$ [10,1000] ms;
- U_i task util. by Uunifast*;
- 1000 systems per U set util.

Cache related:
- N = 512 cache sets, BRT = 8µs;
- U^C = 4 total cache utilization (total # of ECBs is 512 x 4);
- U_i^C task cache util. by UUnifast;
- 40% of ECBs are UCBs.

FPTS *significantly* improves schedulability ratio!

*E. Bini and G. Buttazzo, RTSJ, 2005.

[11] R.J. Bril et al, RTSJ, 2017.

Basic LP-FPS schemes

2. **Deferred preemption: two flavors**
   a) **Scheduling algorithm:**
      - **floating non-pre-emptive region (NPR);**
   b) **Task τ_i:**
      - **each job *is a sequence/DAG of* m_i NPRs (FPDS)**
        - $\forall_i\, m_i = 1 \Rightarrow$ **FPNS**

Relative strength – I

1. **FPTS:**
   - ***Advantages*:**
     - **Described in standards (e.g. OSEK/AUTOSAR);**
     - **Supported by COTS RTOS;**
     - ***Requires no changes to the code* (FPPS & FPNS);**
   - ***Disadvantages*:**
     - **(Limited) arbitrary pre-emptions...**
     - **Open problem: Computational tractable method to determine optimal priorities and pre-emption thresholds of tasks (even *without* CRPD).**

Relative strength – II

2. FPDS

- *Advantages* (compared to FPTS):
  - Less preemption costs;
  - Higher predictability (preemption points known);
  - Higher schedulability ratio;
- *Disadvantages*:
  - Hardly supported by a COTS RTOS (if at all);
  - Visible to a programmer (?);
  - Partially solved: Preemption point placement and CRPD.
  - Open problem: *Optimal* priority assignment, preemption point placement with CRPD.

[29] B. Peng et al., ECRTS, 2014
[17] J. Cavicchio et al., ECRTS, 2015.

Relative strength – III

- **Conclusion on basic LP-FPS models**
  - **Two main (incomparable) alternatives: FPTS & FPDS**
  - **FPDS (versus FPTS):**
    - **highest schedulability ratio;**
    - **generates less preemptions;**
    - **most predictable for estimating preemption costs**
      - **because the number of preemptions and their positions are fixed and known from the code;**
    - **placement of pre-emption points only partially solved;**
    - **typically not supported by COTS RTOS.**

Advanced LP-FPS schemes

- **FPTS and FPDS are orthogonal**
  - **FPTS (scheduling):**
    - **pre-emption threshold per task: θ_i, where $\pi_i \leq \theta_i \leq \pi_1$;**
  - **FPDS (task τ_i):**
    - **every job of is a sequence of m_i sub-jobs (NPRs).**
- **Advanced LP-FPS schemes (FPTS ⊗ FPDS):**
  - **Preemption threshold**
    - **per *sub-job*: $\theta_{i,k}$, where $\pi_i \leq \theta_{i,k} \leq \pi_1$;**
    - **per *pre-emption point*: $\dot{\theta}_{i,k}$, where $\pi_i \leq \dot{\theta}_{i,k} \leq \pi_1$**
      - **superseding θ_i.**

Generalization graph for FPS algorithms

Job as a sequence of m_i sub-jobs.

	$m_i = 1$	$m_i \geq 1$		
		$m_i > 1 \Rightarrow \theta^{\bullet}_{i,a} = \pi_i$		
$\theta_{i,k} = \pi_i$	**FPPS** ←	**FPPS⁺**		
$\theta_{i,k} = \pi_1$	**FPNS** ←	**FPDS**		

[15] A. Burns, in *Advances in Real-Time Systems*, 1994

[13] R. Bril, J. Lukkien, and W. Verhaegh, ECRTS, 2007.

Generalization graph for FPS algorithms

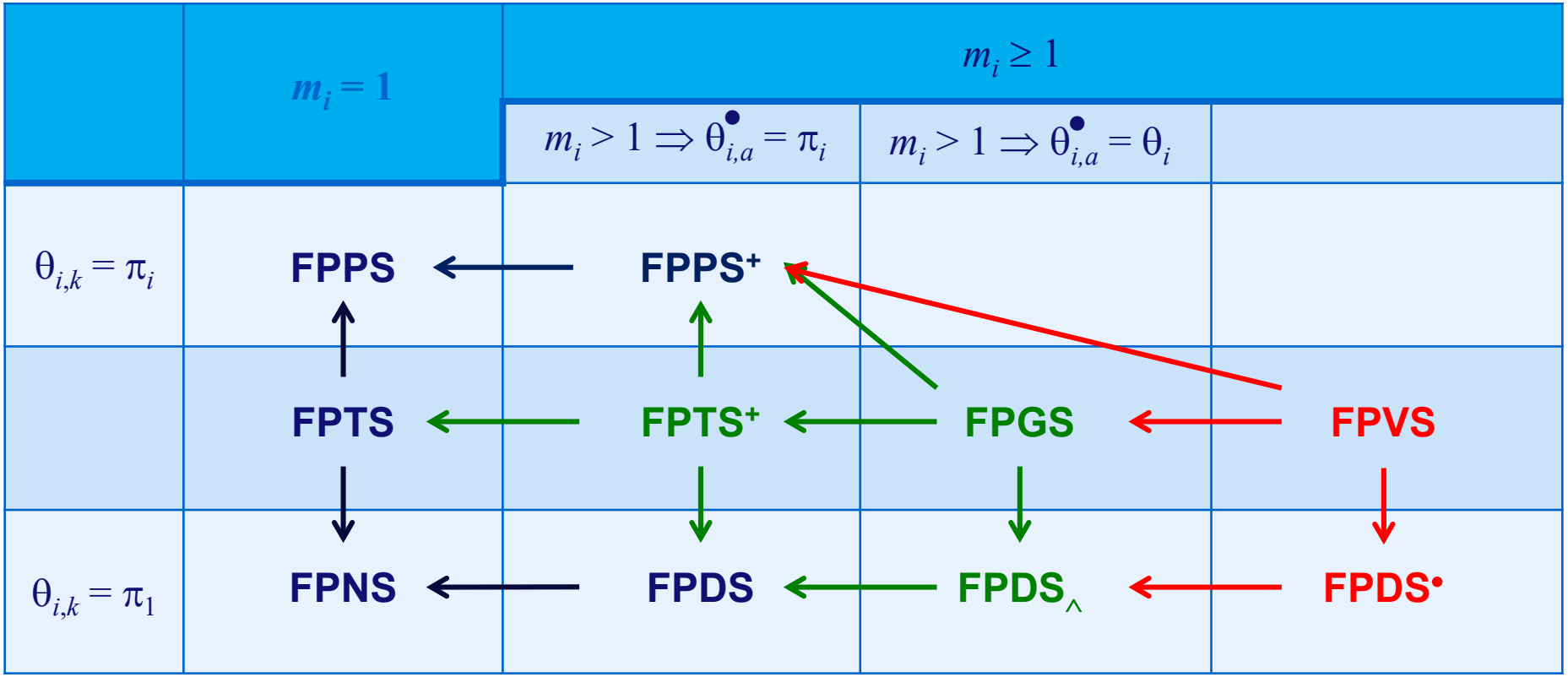
Preemption threshold per task.

	$m_i = 1$	$m_i \geq 1$		
		$m_i > 1 \Rightarrow \theta_{i,a}^{\bullet} = \pi_i$		
$\theta_{i,k} = \pi_i$	**FPPS**	**FPPS⁺**		
	FPTS			
$\theta_{i,k} = \pi_1$	**FPNS**	**FPDS**		

[33] Y. Wang and M. Saksena, RTCSA, 1999.
[30] J. Regehr, RTSS, 2002.

Generalization graph for FPS algorithms

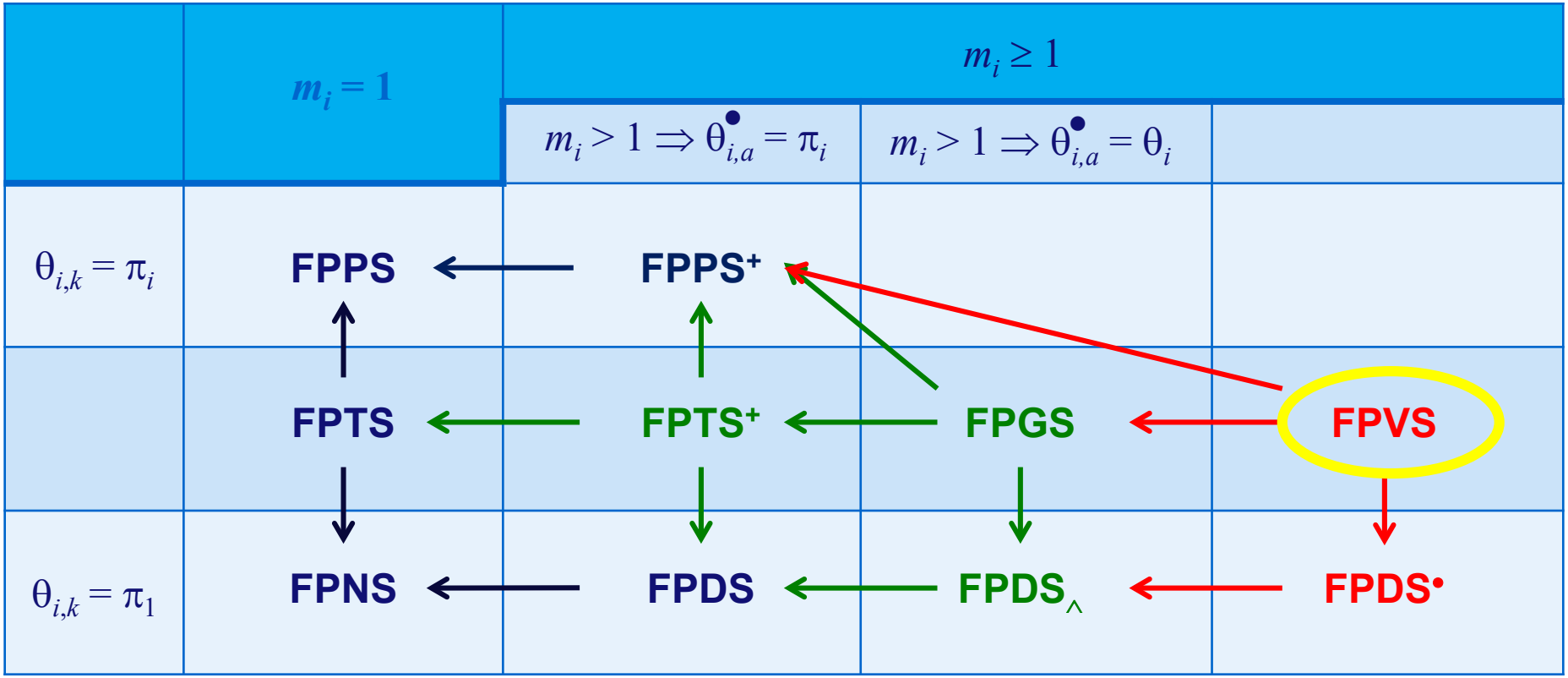
Preemption threshold per sub-job and per preemption point.



[14] R.J. Bril et al. RTNS, 2013.

Generalization graph for FPS algorithms

Preemption threshold per sub-job and per preemption point.

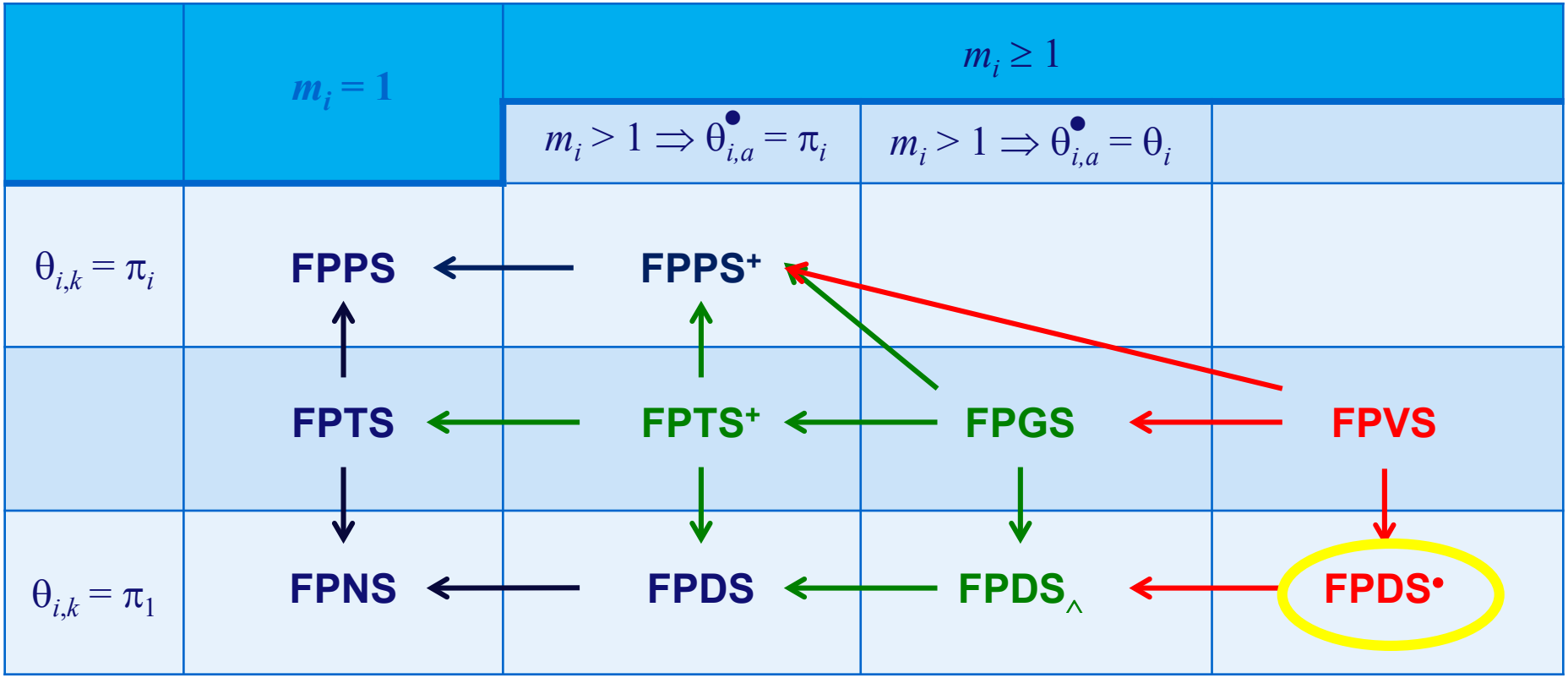


FPVS: *Pro*: highest schedulability ratio;
Cons: potential of arbitrary preemptions;
Cons: potential of higher memory requirements.

Technische Universiteit Eindhoven University of Technology

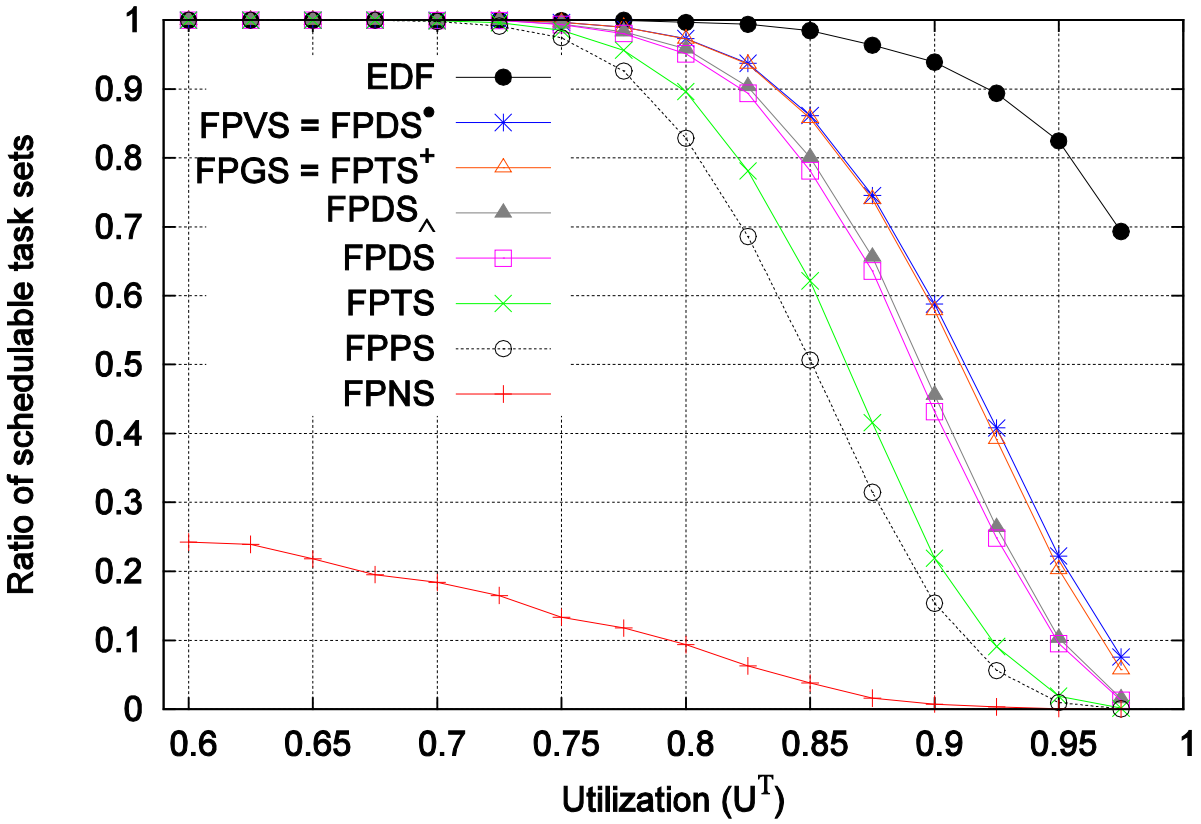
Generalization graph for FPS algorithms

Preemption threshold per sub-job and per preemption point.



FPDS•: *Pro*: high schedulability ratio;
Pro: predictable preemptions costs;
Pro: predictable memory requirements.

Schedulability Ratio



8 tasks, 5.000 task sets, DMPO

$T_i \in$ [100, 10.000] (uniform), U_i by UUnifast ($\Rightarrow C_i$),

$D_i \in$ [0.5(T_i + C_i), T_i] (uniform);

Optimal Scheduling for FPVS

- **Fundamental results for FPVS:**
  1. **Given priorities for all tasks, an algorithm exists to determine optimal settings for $C_{i,a}$, $\theta_{i,a}$, $\theta^{\bullet}_{i,a}$ using n schedulability tests.**
  2. **Open problems:**
     - ***Optimal* priorities and settings for $C_{i,a}$, $\theta_{i,a}$, $\theta^{\bullet}_{i,a}$.**
     - **Analysis including CRPD**
       - **Optimal…**

[14] R.J. Bril et al., RTNS, 2013.

Support for advanced LP-FPS

- **OS-level:**
  - **Some work on prototyping (FPDS [7], FPVS, FPDS•[6]);**
  - **Shall be *efficient,* however.**
- **Additional tooling:**
  - **Preemption-point placement…**

[7] M. Bergsma et al., OSPERT, 2009.
[6] M. Becker et al. SIES, 2015.

Summary of Open Problems for LP-FPS

- **Theory:**
  - **Stack requirements:**
    - **for generalizations of FPTS and FPDS...**
  - **Scheduling:**
    - **a computational tractable OPTA for FPTS;**
      - **...and generalizations of FPTS and FPDS...**
    - **analysis including CRPD for generalizations**
      - **…and optimal settings;**
- **Practice**
  - **OS support;**
  - **Tool support.**

Conclusion

- **Today, FPPS is the de facto standard in industry, but not cost-effective due to *arbitrary* preemptions.**
- **Compared to FPPS, LP-FPS may significantly improve *cost-effectiveness* of systems.**
- **LP-FPS is hardly supported by COTS RTOS (if at all).**
- **Many open research questions**
  - **Even for "FPPS with CRPD";**
  - **Especially for LP-FPS.**
- **Additional tool-support needed as well.**
- **Recommended: support for LP-FPS in NGOSCPS.**

Questions

- **Thank you for you attention...**