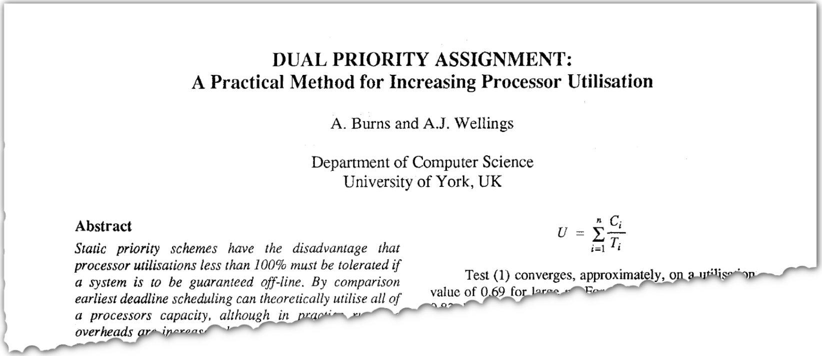
Dual Priority Scheduling is Not Optimal

Pontus Ekberg
Uppsala University

ECRTS 2019
Stuttgart, Germany

Dual priority scheduling?

Dual priority scheduling?



DUAL PRIORITY ASSIGNMENT:
A Practical Method for Increasing Processor Utilisation

A. Burns and A.J. Wellings

Department of Computer Science
University of York, UK

Abstract

Static priority schemes have the disadvantage that processor utilisations less than 100% must be tolerated if a system is to be guaranteed off-line. By comparison earliest deadline scheduling can theoretically utilise all of a processors capacity, although in

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Test (1) converges, approximately, on a

value of 0.69 for

(Euromicro Workshop on Real-Time Systems, 1993)

How does it work?

Assumptions

- Implicit deadline periodic tasks
- Single preemptive processor

How does it work?

Assumptions

- Implicit deadline periodic tasks
- Single preemptive processor

Configurations

For each task $\tau_i = (e_i, p_i)$, assign

- a phase change point $\delta_i \in \{0, \ldots, p_i\}$
- two priority levels: π_i^1 and π_i^2

How does it work?

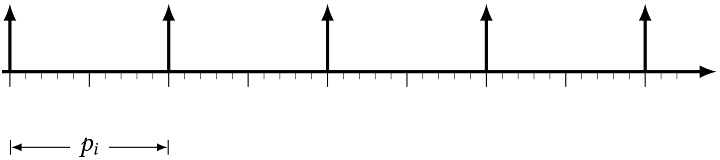
Assumptions

- Implicit deadline periodic tasks
- Single preemptive processor

Configurations

For each task $\tau_i = (e_i, p_i)$, assign

- a phase change point $\delta_i \in \{0, \ldots, p_i\}$
- two priority levels: π_i^1 and π_i^2



How does it work?

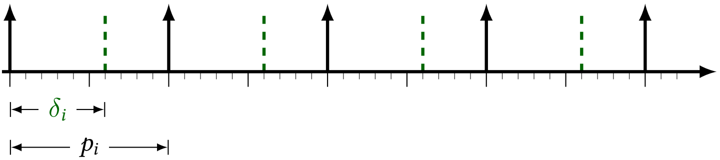
Assumptions

- Implicit deadline periodic tasks
- Single preemptive processor

Configurations

For each task $\tau_i = (e_i, p_i)$, assign

- a phase change point $\delta_i \in \{0, \ldots, p_i\}$
- two priority levels: π_i^1 and π_i^2



How does it work?

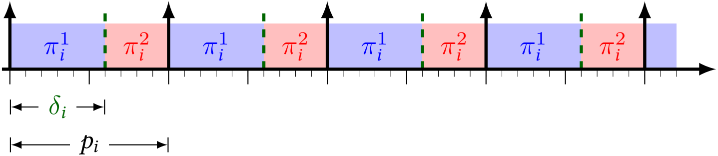
Assumptions

- Implicit deadline periodic tasks
- Single preemptive processor

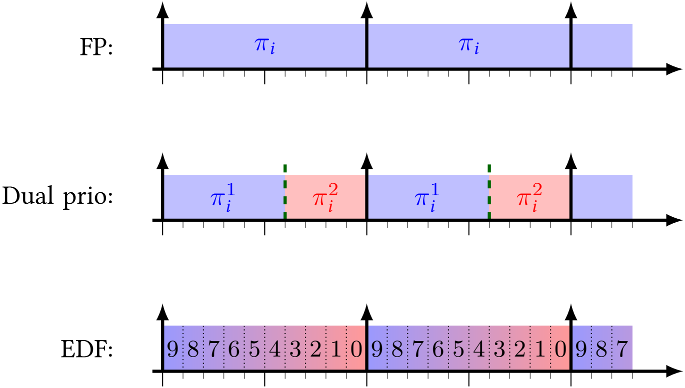
Configurations

For each task $\tau_i = (e_i, p_i)$, assign

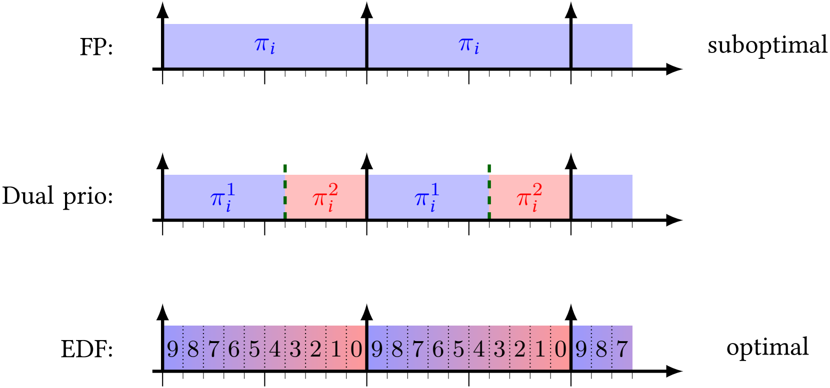
- a phase change point $\delta_i \in \{0, \ldots, p_i\}$
- two priority levels: π_i^1 and π_i^2



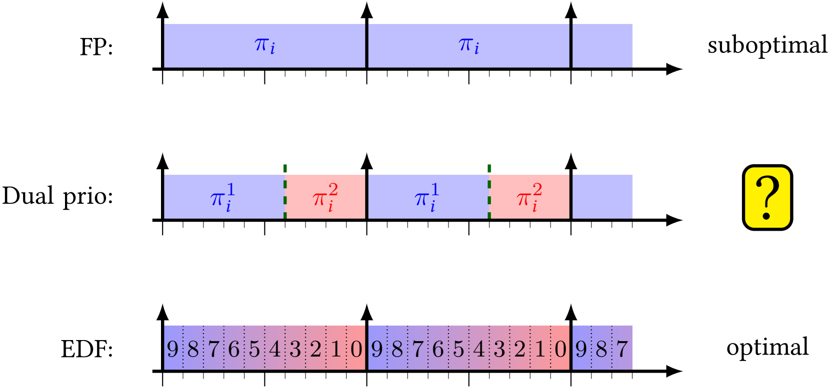
Comparing with FP and EDF



Comparing with FP and EDF



Comparing with FP and EDF



Is dual priority scheduling optimal?

priorities be found.

The answer to the first takes the form of a conjecture:

Conjecture C1

For any task set with total utilisation less than or equal to 100% there exists a dual priority assignment that will meet all deadlines.

Analysis of this conjecture in given in section 4.

τ_3				
τ_4	20	6		
Total Utilisation		100%		
LCM		240		

Table 2: Task Set E2

With no task having a second phase, the system is not

Conjecture 1 (Burns and Wellings, 1993)

Dual priority scheduling is *optimal* for implicit deadline periodic tasks.

(Sadly not)

Why was this a difficult conjecture?

Why was this a difficult conjecture?

Disproving the conjecture

Find a task set that is

- feasible ($U \leqslant 1$), and
- not dual priority schedulable.

Why was this a difficult conjecture?

Disproving the conjecture

Find a task set that is

- feasible ($U \leqslant 1$), and
- not dual priority schedulable.

Easy!

Why was this a difficult conjecture?

Disproving the conjecture

Find a task set that is

- feasible ($U \leqslant 1$), and
- not dual priority schedulable.

Easy!

But in practice…

- Almost all task sets are schedulable
- Evaluating the schedulability can be *very* costly

Why was this a difficult conjecture?

Disproving the conjecture

Find a task set that is

- feasible ($U \leqslant 1$), and
- not dual priority schedulable.

Easy!

But in practice...

- Almost all task sets are schedulable
- Evaluating the schedulability can be *very* costly

Schedulability test

For every configuration (setting of $\pi_i^1, \pi_i^2, \delta_i$), simulate the hyper-period until a deadline miss.

How many configurations are there?

Assumptions

- All priority levels are unique
  - $\Rightarrow$ A total of $(2n)!$ permutations
- Phase change points (δ_i) are integer
  - $\Rightarrow$ A total of $\prod_{i=1}^{n}(p_i + 1)$ combinations

How many configurations are there?

Assumptions

- All priority levels are unique
  - $\Rightarrow$ A total of $(2n)!$ permutations
- Phase change points (δ_i) are integer
  - $\Rightarrow$ A total of $\prod_{i=1}^{n}(p_i + 1)$ combinations

A task set $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ has

$$(2n)! \times \prod_{i=1}^{n}(p_i + 1)$$

distinct configurations.

A counterexample

Not dual priority schedulable

	e_i	p_i
τ_1	8	19
τ_2	13	29
τ_3	9	151
τ_4	14	197

hyper-period: 16 390 597
utilization: ~ 0.9999971

A counterexample

Not dual priority schedulable

	e_i	p_i
τ_1	8	19
τ_2	13	29
τ_3	9	151
τ_4	14	197

hyper-period: 16 390 597
utilization: ~ 0.9999971

$$\#\text{configurations} \;=\; (2n)! \times \prod_{i=1}^{n}(p_i + 1) \;=\; 728\,082\,432\,000$$

A counterexample

Not dual priority schedulable

	e_i	p_i
τ_1	8	19
τ_2	13	29
τ_3	9	151
τ_4	14	197

hyper-period: 16 390 597
utilization: ~ 0.9999971

$$\#\text{configurations} = (2n)! \times \prod_{i=1}^{n} (p_i + 1) = 728\,082\,432\,000$$

Simulating the full hyper-period for all configurations would take *hundreds of years* on my computer.

The saving grace

Most configurations lead to a deadline miss very early.

The saving grace

Most configurations lead to a deadline miss very early.

For the previous counterexample, less than

$$0.00019\ \%$$

of the hyper-period is simulated on average.

The saving grace

Most configurations lead to a deadline miss very early.

For the previous counterexample, less than

$$0.00019\ \%$$

of the hyper-period is simulated on average.

Simulation time is ~ 2.5 days on an office computer.

Recognizing the needle: RM+RM

$$\#\text{configurations} \;=\; (2n)! \times \prod_{i=1}^{n}(p_i + 1)$$

Recognizing the needle: RM+RM

$$\#\text{configurations} \; = \; \cancel{(2n)!} \times \prod_{i=1}^{n}(p_i + 1)$$

Recognizing the needle: RM+RM

$$\#\text{configurations} \;=\; \cancel{(2n)!} \times \prod_{i=1}^{n}(p_i + 1)$$

Definition: RM+RM

A dual priority configuration is called *RM+RM* if

1. phase 1 priorities (π_i^1) are RM
2. phase 2 priorities (π_i^2) are RM
3. $\max_i\{\pi_i^2\} \leqslant \min_i\{\pi_i^1\}$

Recognizing the needle: RM+RM

$$\#\text{configurations} \;=\; \cancel{(2n)!} \times \prod_{i=1}^{n}(p_i+1)$$

Definition: RM+RM

A dual priority configuration is called *RM+RM* if

1. phase 1 priorities (π_i^1) are RM
2. phase 2 priorities (π_i^2) are RM
3. $\max_i\{\pi_i^2\} \leqslant \min_i\{\pi_i^1\}$

Conjecture 2 (George et al., 2014)

RM+RM is an optimal choice of priorities.

(Sadly not, even considering only point 1 above)

Recognizing the needle: FDMS (Fautrel et al., 2018)

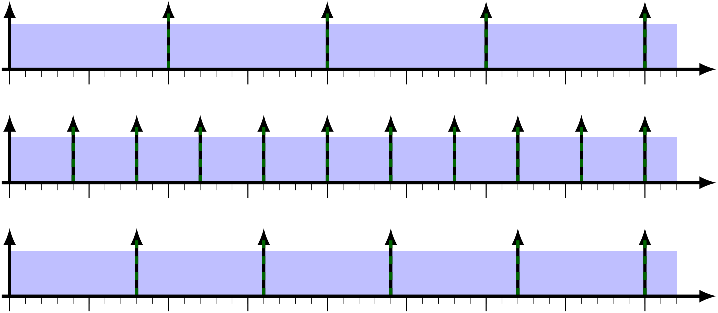
$$\#\text{configurations} \;=\; \cancel{(2n)!} \times \prod_{i=1}^{n}(p_i + 1)$$

Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \; = \; (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1})$$

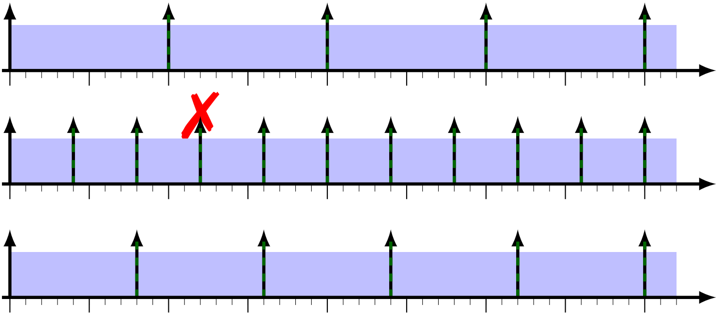
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i +} 1)$$



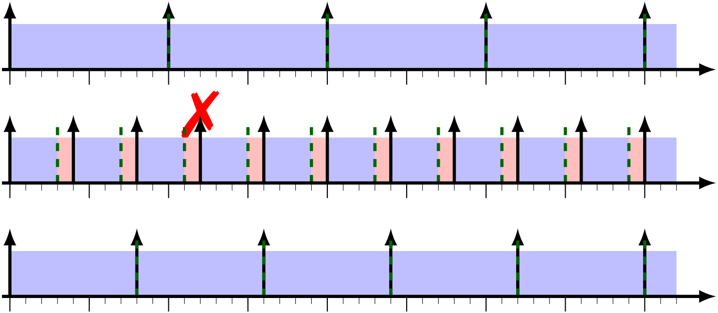
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \; = \; (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i} + 1)$$



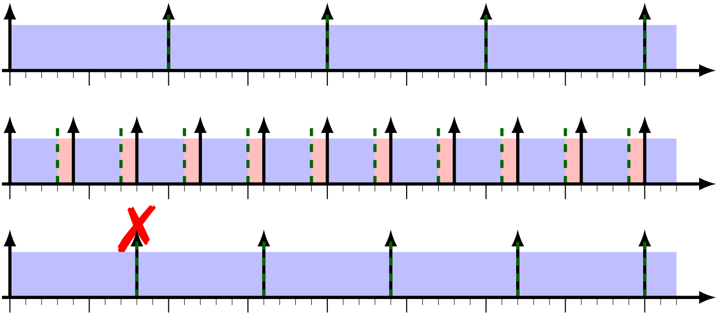
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} = (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1)}$$



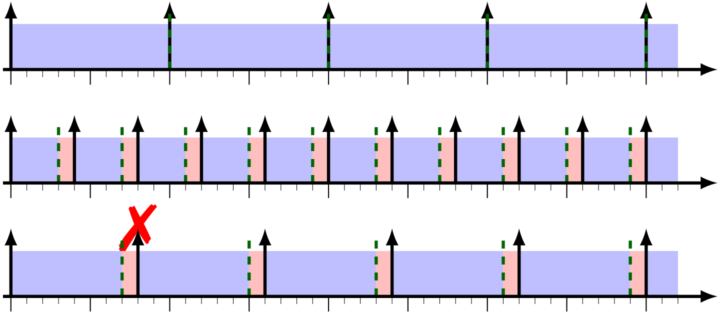
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i} + 1)$$



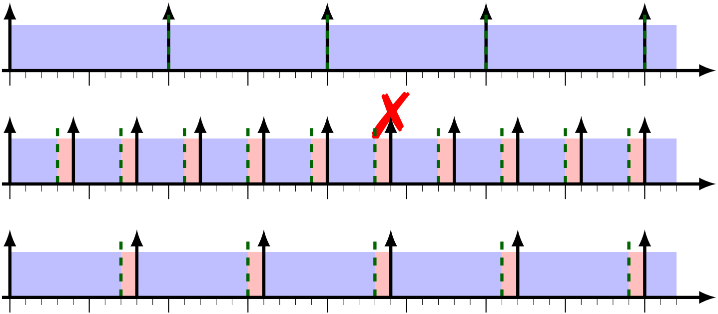
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} = (2\cancel{n})! \times \prod_{i=1}^{n}\cancel{(p_i + 1)}$$



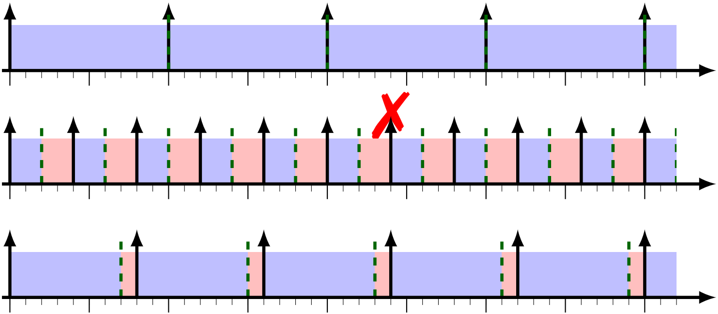
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} = (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1)}$$



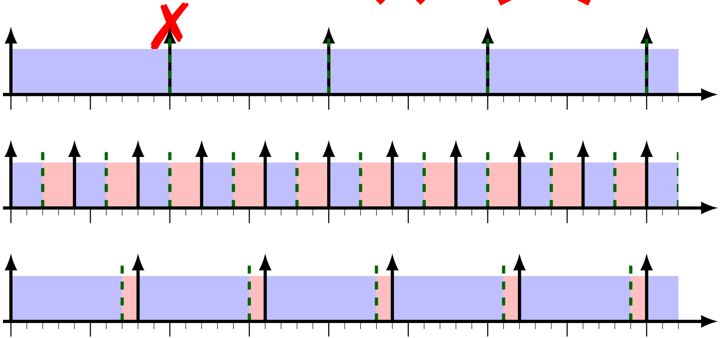
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1)}$$



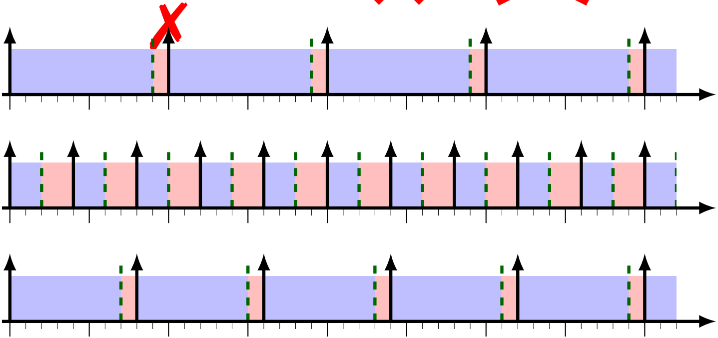
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} = (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1)}$$



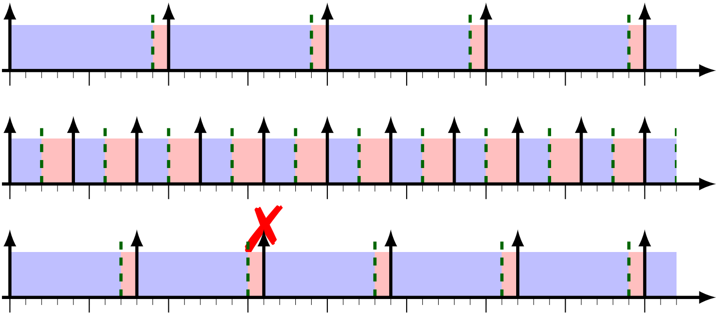
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1)}$$



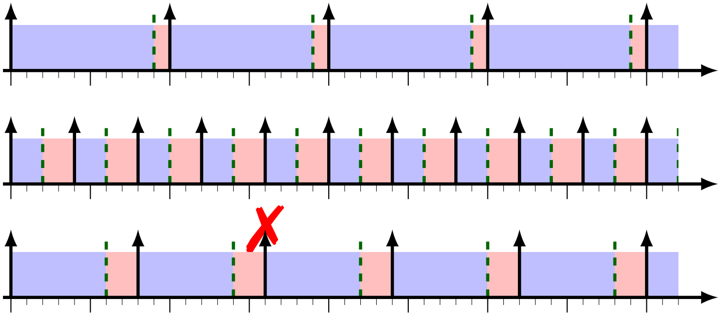
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (\cancel{2n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1)}$$



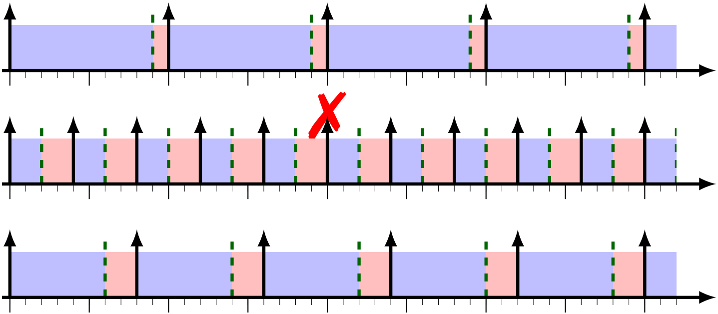
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (2\cancel{n})! \times \prod_{i=1}^{n}(\cancel{p_i}+1)$$



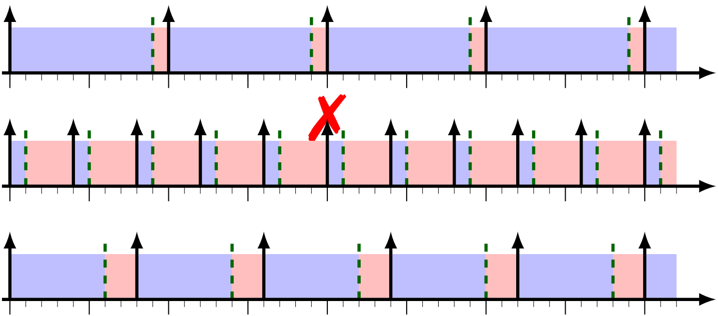
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i} + 1)$$



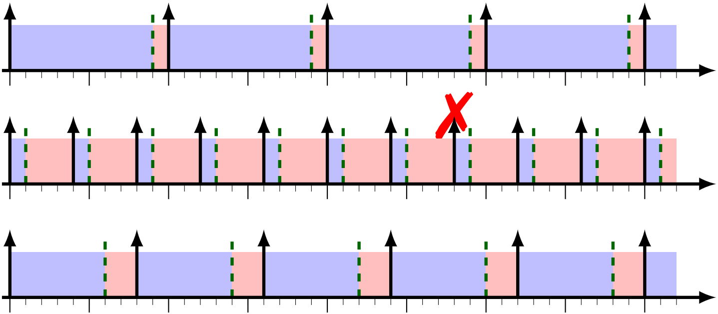
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (\cancel{2n})! \times \prod_{i=1}^{n} \cancel{(p_i +} 1)$$



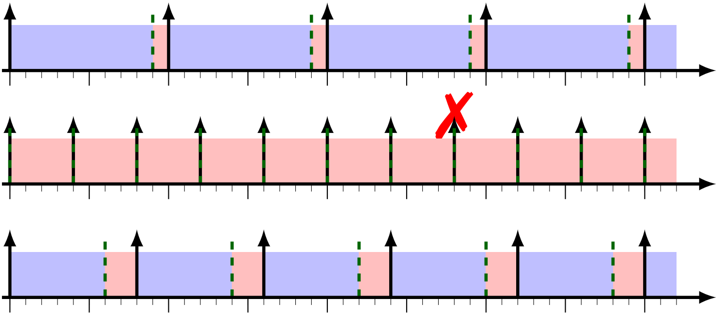
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} \;=\; (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1)}$$



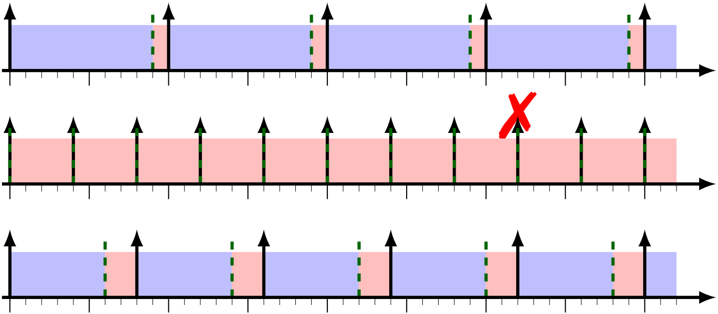
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} = (2\cancel{n})! \times \prod_{i=1}^{n} \cancel{(p_i + 1)}$$



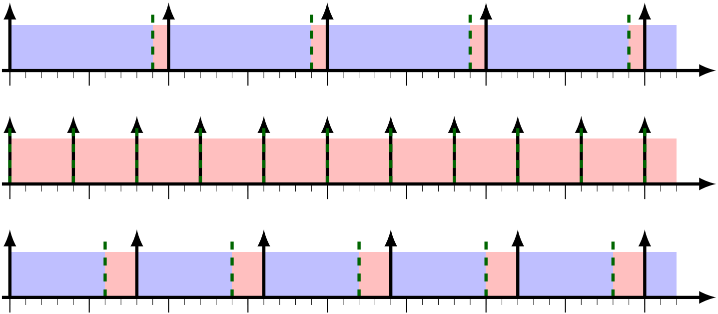
Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} = (2\cancel{n})! \times \prod_{i=1}^{n}\cancel{(p_i + 1)}$$



Recognizing the needle: FDMS (Fautrel et al., 2018)

$$\#\text{configurations} = \cancel{(2n)!} \times \cancel{\prod_{i=1}^{n}(p_i+1)}$$



Conjecture 3 (Fautrel et al., 2018)

FDMS always finds optimal phase change points.

(Sadly not)

Recognizing the needle

A simple schedulability analysis strategy

1. Try *RM+RM* priorities with *FDMS*
2. If not successful, check configurations exhaustively

Doing 1 is *much* faster than 2, and works most of the time.

Where is my haystack?

Where is my haystack?

An ad hoc search space

- 4 tasks
- Utilization $\in [0.99999, 1]$
- Periods chosen from the first 100 primes
- $\prod_{i=1}^{n} p_i \leqslant 35\,000\,000$

Where is my haystack?

An ad hoc search space

- 4 tasks
- Utilization $\in [0.99999, 1]$
- Periods chosen from the first 100 primes
- $\prod_{i=1}^{n} p_i \leqslant 35\,000\,000$

Random task sets were tested from this search space until an unschedulable one was found. This is the breakdown:

	# task sets	% of explored search space
Schedulable with RM+RM using FDMS	129 823	$\sim 99.67\%$
Schedulable with other configurations	431	$\sim 0.33\%$
Unschedulable	1	$\sim 0.0008\%$

So dual priority scheduling is no good?

So dual priority scheduling is no good?

Fact

It took *26 years* and evaluating *millions of task sets* to find a single unschedulable one.

So dual priority scheduling is no good?

Fact

It took *26 years* and evaluating *millions of task sets* to find a single unschedulable one.

But perhaps it behaves worse for larger task sets.

So dual priority scheduling is no good?

Fact

It took *26 years* and evaluating *millions of task sets* to find a single unschedulable one.

But perhaps it behaves worse for larger task sets.

To fully exploit the apparent near-optimality we need efficient tests and methods for finding the right parameters.

Open problems

Open problem 1

Can we efficiently determine if there exists a schedulable configuration?

Open problems

Open problem 1

Can we efficiently determine if there exists a schedulable configuration?

In PSPACE, no lower bounds known

Open problems

Open problem 1

Can we efficiently determine if there exists a schedulable configuration?

In PSPACE, no lower bounds known

Open problem 2

If yes, can we efficiently find it?

Open problems

Open problem 1

Can we efficiently determine if there exists a schedulable configuration?

In PSPACE, no lower bounds known

Open problem 2

If yes, can we efficiently find it?

Open problem 3

Can we efficiently evaluate a given configuration?

Open problems

Open problem 1

Can we efficiently determine if there exists a schedulable configuration?

In PSPACE, no lower bounds known

Open problem 2

If yes, can we efficiently find it?

Open problem 3

Can we efficiently evaluate a given configuration?

In PSPACE, weakly NP-hard

Open problems

Open problem 1

Can we efficiently determine if there exists a schedulable configuration?

In PSPACE, no lower bounds known

Open problem 2

If yes, can we efficiently find it?

Open problem 3

Can we efficiently evaluate a given configuration?

In PSPACE, weakly NP-hard

Simulation only works for periodic tasks (George et al., 2014)

Open problems

Open problem 4

What is the utilization bound?

Open problems

Open problem 4

What is the utilization bound?

Must be in the interval $[\ln(2), 1)$

Open problems

Open problem 4

What is the utilization bound?

Must be in the interval $[\ln(2), 1)$

Open problem 5

Is k-priority scheduling optimal for some constant k?

Open problems

Open problem 4

What is the utilization bound?

Must be in the interval $[\ln(2), 1)$

Open problem 5

Is k-priority scheduling optimal for some constant k?

It is for n-priority scheduling, where $n = |\mathcal{T}|$ (Pathan, 2015)

Open problems

Open problem 4

What is the utilization bound?

Must be in the interval $[\ln(2), 1)$

Open problem 5

Is k-priority scheduling optimal for some constant k?

It is for n-priority scheduling, where $n = |\mathcal{T}|$ (Pathan, 2015)

Open problem 6

What about rational phase change points?

Open problems

Open problem 4

What is the utilization bound?

Must be in the interval $[\ln(2), 1)$

Open problem 5

Is k-priority scheduling optimal for some constant k?

It is for n-priority scheduling, where $n = |\mathcal{T}|$ (Pathan, 2015)

Open problem 6

What about rational phase change points?

Can't be brute forced!

What if not all priorities are unique?

Precondition

We need to define the tie-breaking rule.

What if not all priorities are unique?

Precondition

We need to define the tie-breaking rule.

But most of them don't make sense!

What if not all priorities are unique?

Precondition

We need to define the tie-breaking rule.

But most of them don't make sense!

Bad news

The number of priority orderings with ties are counted by the *Fubini numbers*, growing faster than the factorials.

What if not all priorities are unique?

Precondition

We need to define the tie-breaking rule.

But most of them don't make sense!

Bad news

The number of priority orderings with ties are counted by the *Fubini numbers*, growing faster than the factorials.

Result

The same counterexample remains unchedulable if priorities can be shared and FIFO or LIFO is used for ties.

(Not in the paper)

Special thanks to

Martina Maggio

Joël Goossens

Artifact evaluators

$\forall$Thank you!

$\Diamond$

$\exists$Questions?