INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES

Scuola Superiore
Sant'Anna

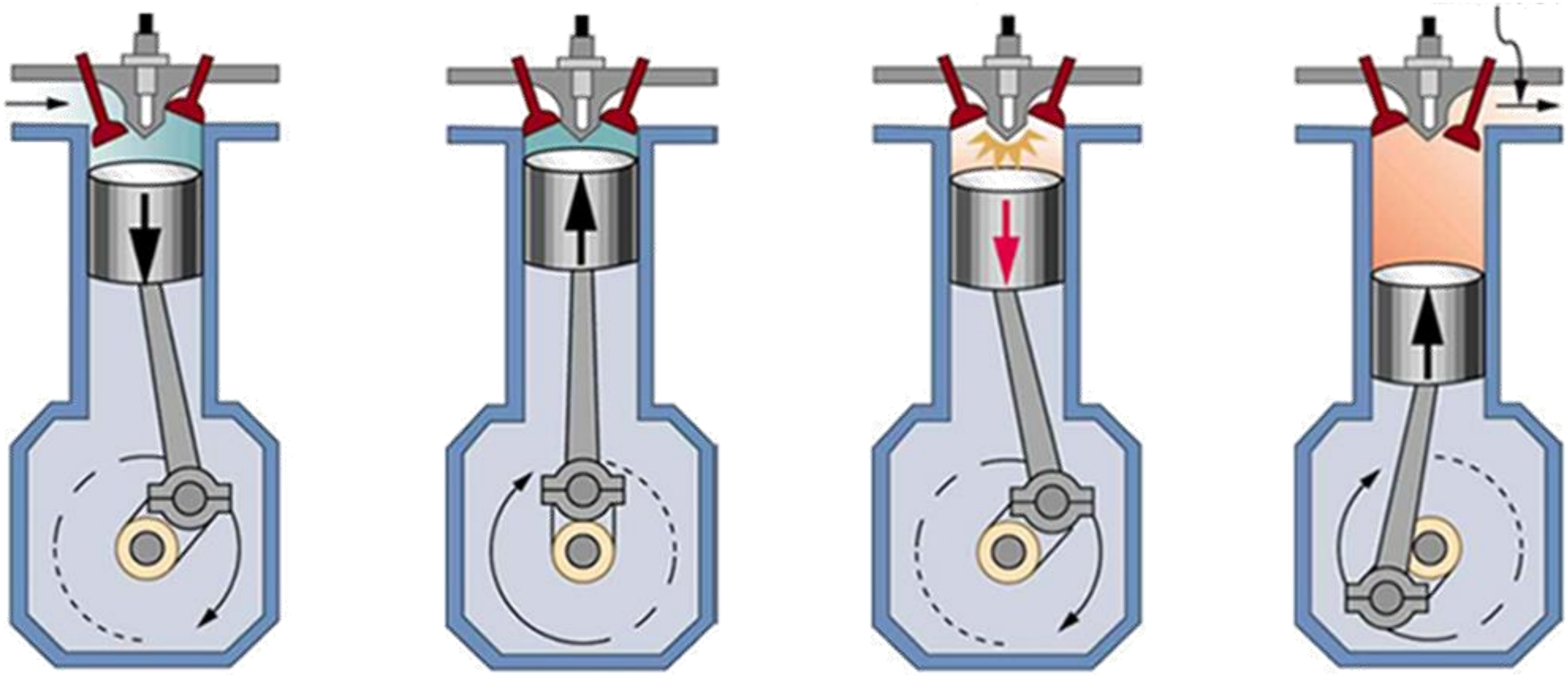
Retis
Real-Time Systems Laboratory

Exact Interference of Adaptive Variable-Rate Tasks Under Fixed-Priority Scheduling

**Alessandro Biondi, Alessandra Melani, Mauro Marinoni,
Marco Di Natale, Giorgio Buttazzo**

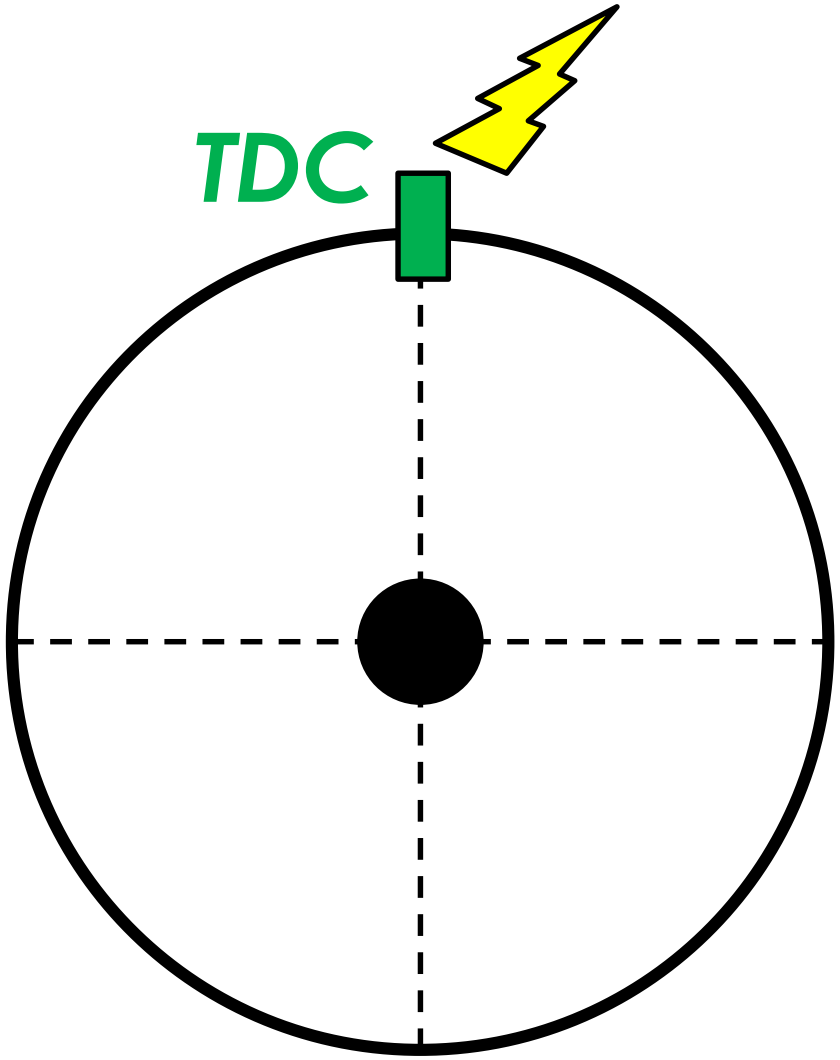
Introduction

- Engine control applications are composed by **Engine-triggered** tasks linked to the rotation of the **crankshaft**



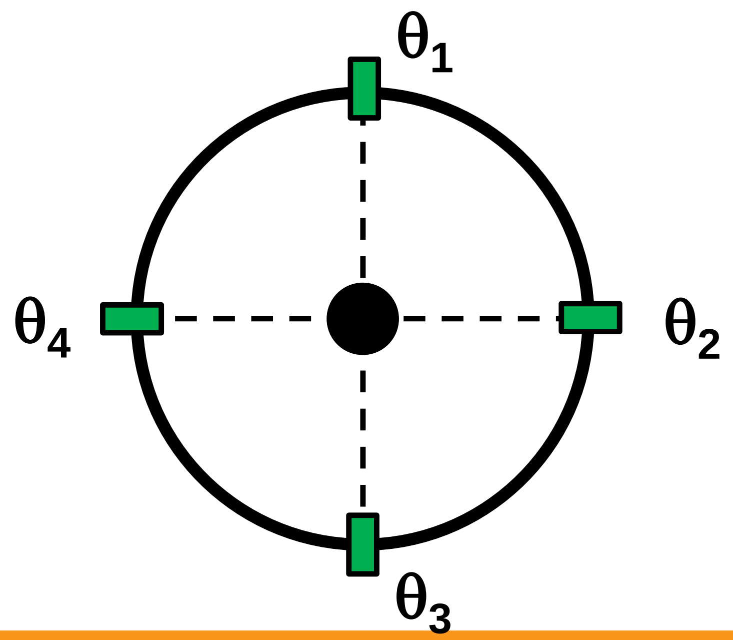
Introduction

- **Engine-triggered** tasks



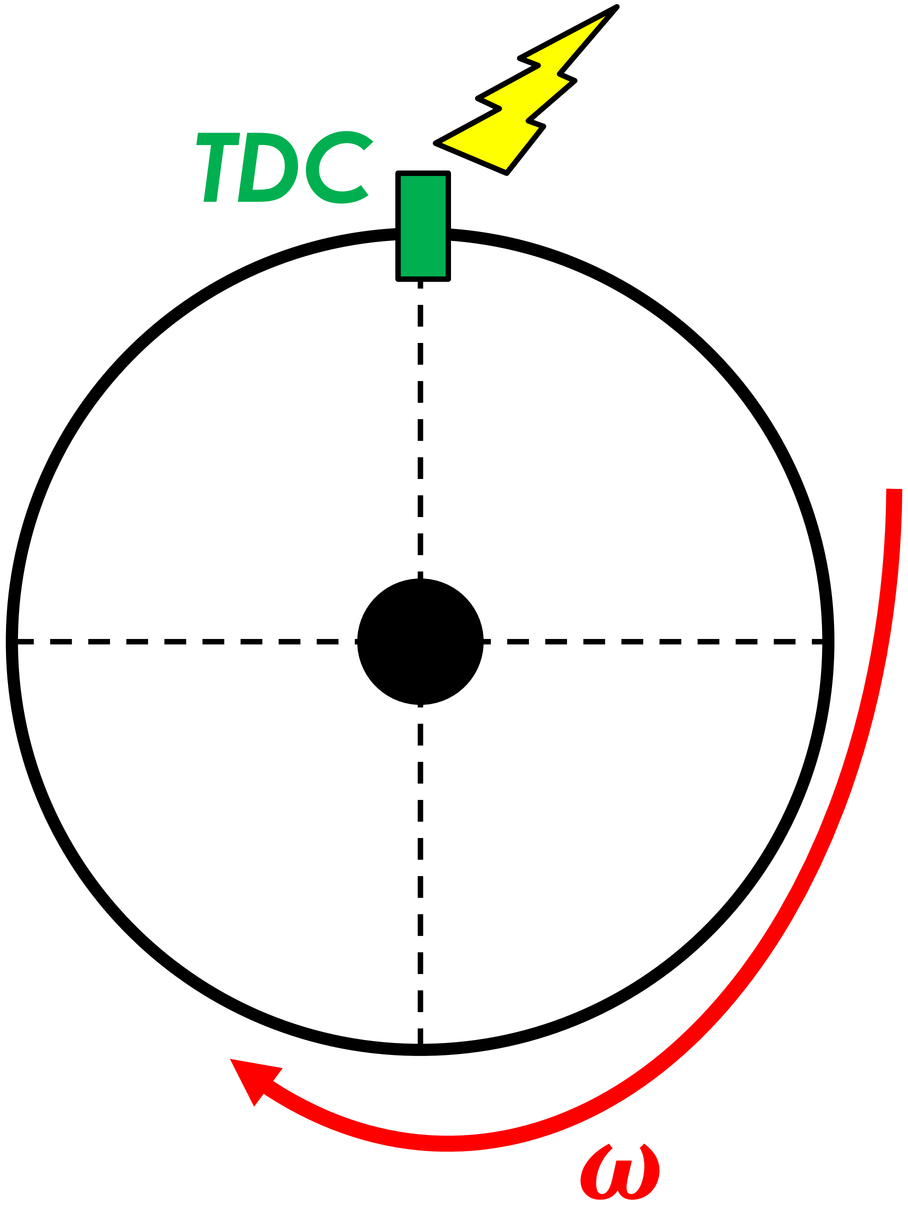
In general:

- The task activation is triggered at specific *rotation angles*



Introduction

- **Engine-triggered** tasks – *single activation per revolution*



Inter-arrival time given a fixed speed ω

$$\boxed{T = \frac{2\pi}{\omega}}$$

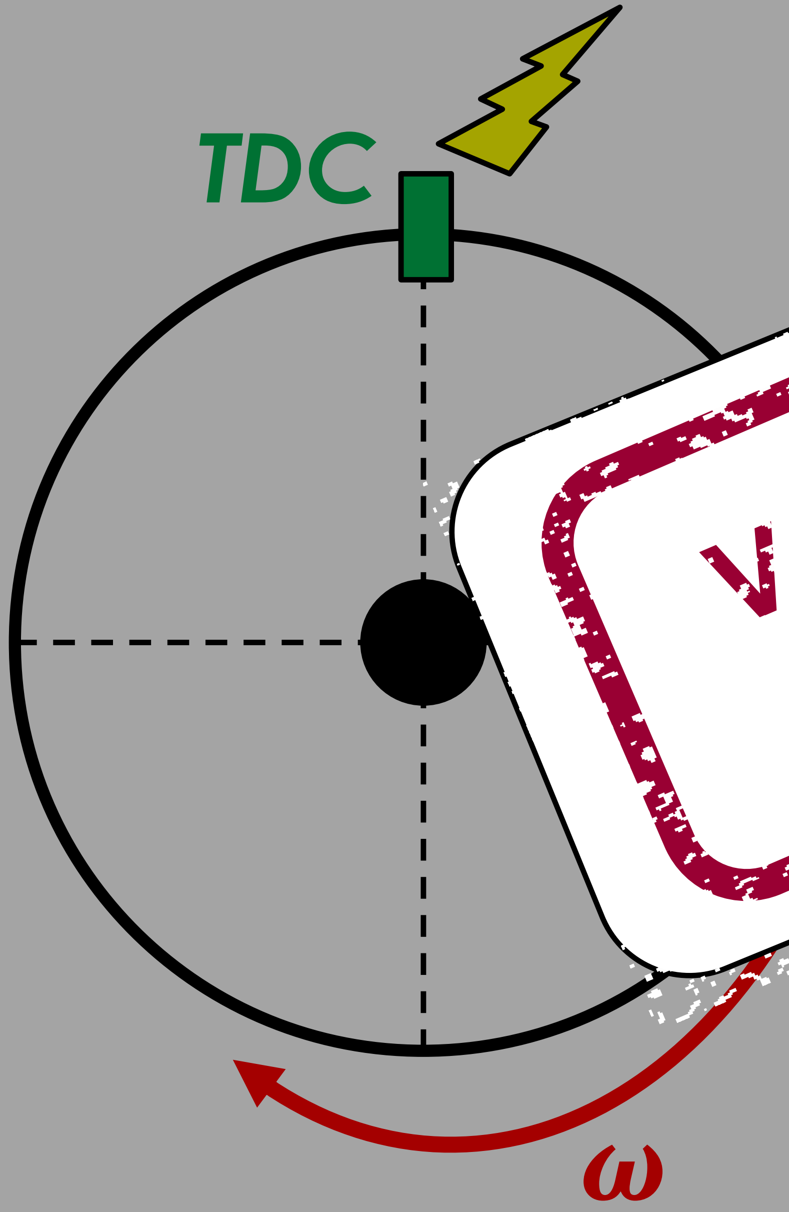
ω^{min} = 500 rpm - ω^{max} = 6500 rpm

⇓ ⇓

T^{max} = **120 ms** - T^{min} ~= **10 ms**

Introduction

- **Engine-triggered** tasks – *single activation per revolution*



VARIABLE-RATE TASKS

Inter-ar e

$$T = \frac{2\pi}{\omega}$$

rpm - ω^{max}= 6500 rpm

T^{max}= 120 ms - T^{min} ~= 10 ms

Introduction

- **High variability** of the inter-arrival time

T^{max}= 120 ms - T^{min} ~= 10 ms

Suppose a fixed WCET C

$$U^{min} = \frac{C}{120\ ms} \xrightarrow{\omega \text{ increases}} U^{max} \sim= \frac{C}{10\ ms}$$

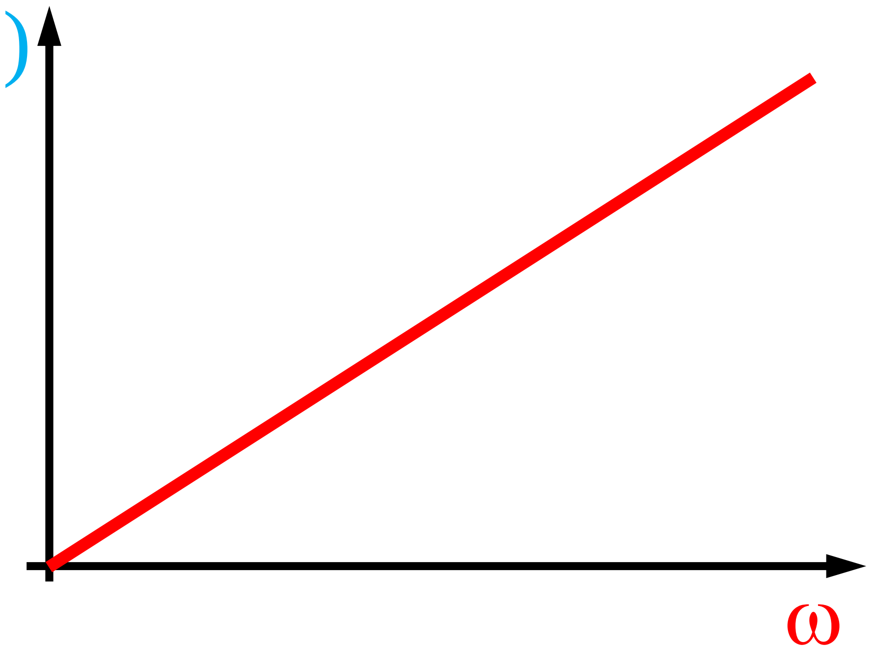
Can be “very low”

Can be “very high”

Introduction

- Hig

$$u(\omega) = \frac{C}{T} = \frac{C}{2\pi}\omega$$



ω increases

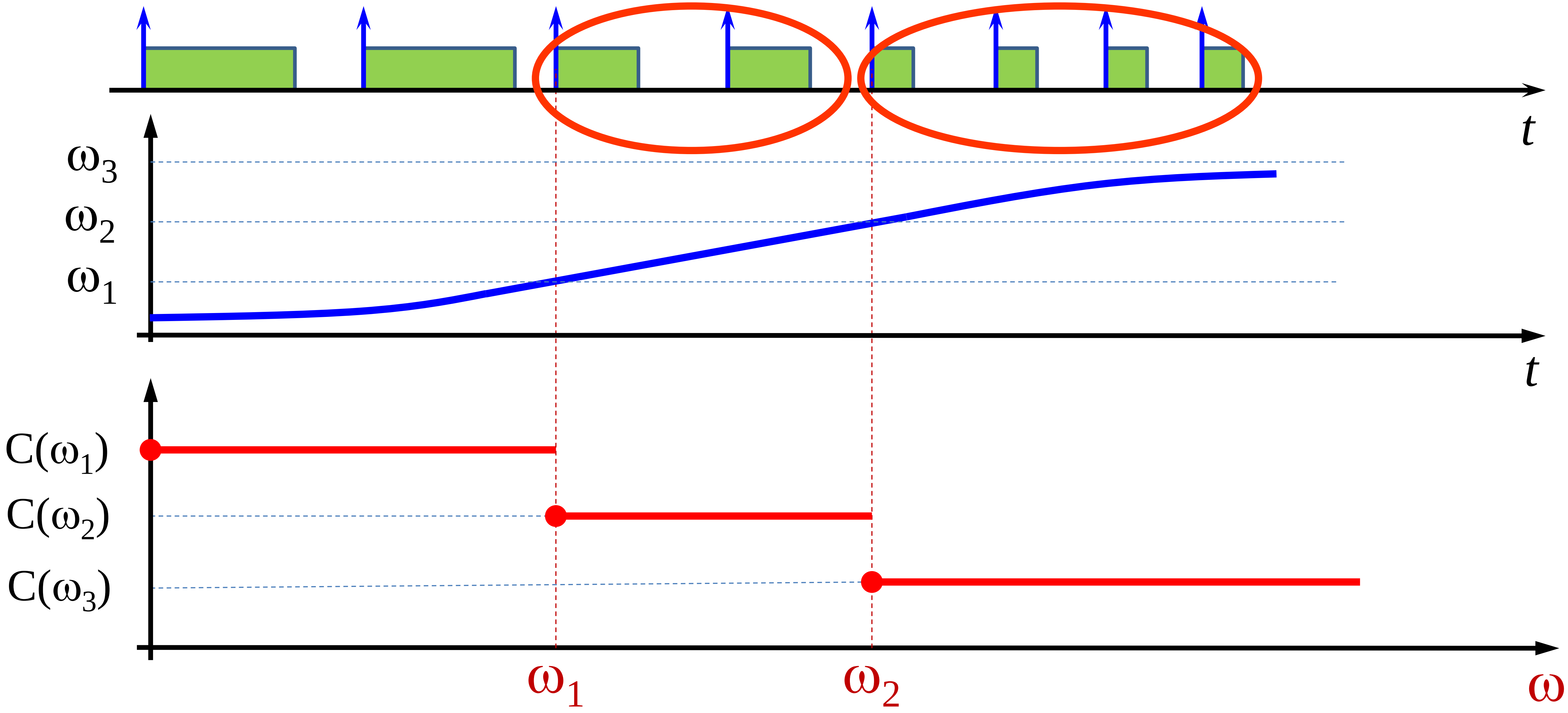
$$U^{min} = \frac{C}{120\ ms} \qquad U^{max} \sim= \frac{C}{10\ ms}$$

Can be "very low"

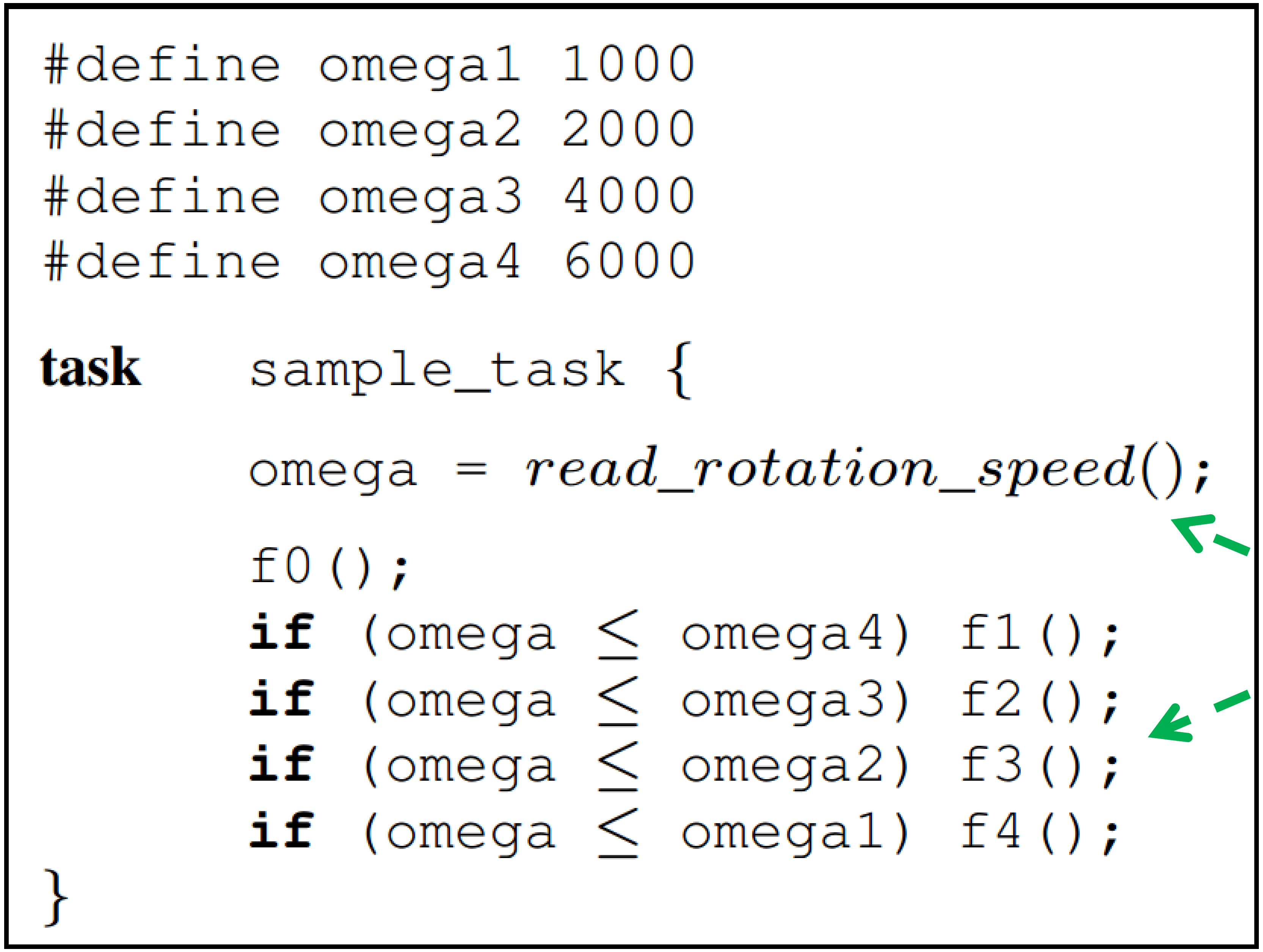
Can be "very high"

Introduction

To prevent **overload** at high rates, certain task functions are *disabled* after given speeds



Adaptive Variable-Rate Tasks



```
#define omega1 1000
#define omega2 2000
#define omega3 4000
#define omega4 6000

task    sample_task {

        omega = read_rotation_speed();

        f0();
        if (omega ≤ omega4) f1();
        if (omega ≤ omega3) f2();
        if (omega ≤ omega2) f3();
        if (omega ≤ omega1) f4();
}
```

Adaptive behavior as a function of the instantaneous engine speed

Adaptive Variable-Rate Tasks

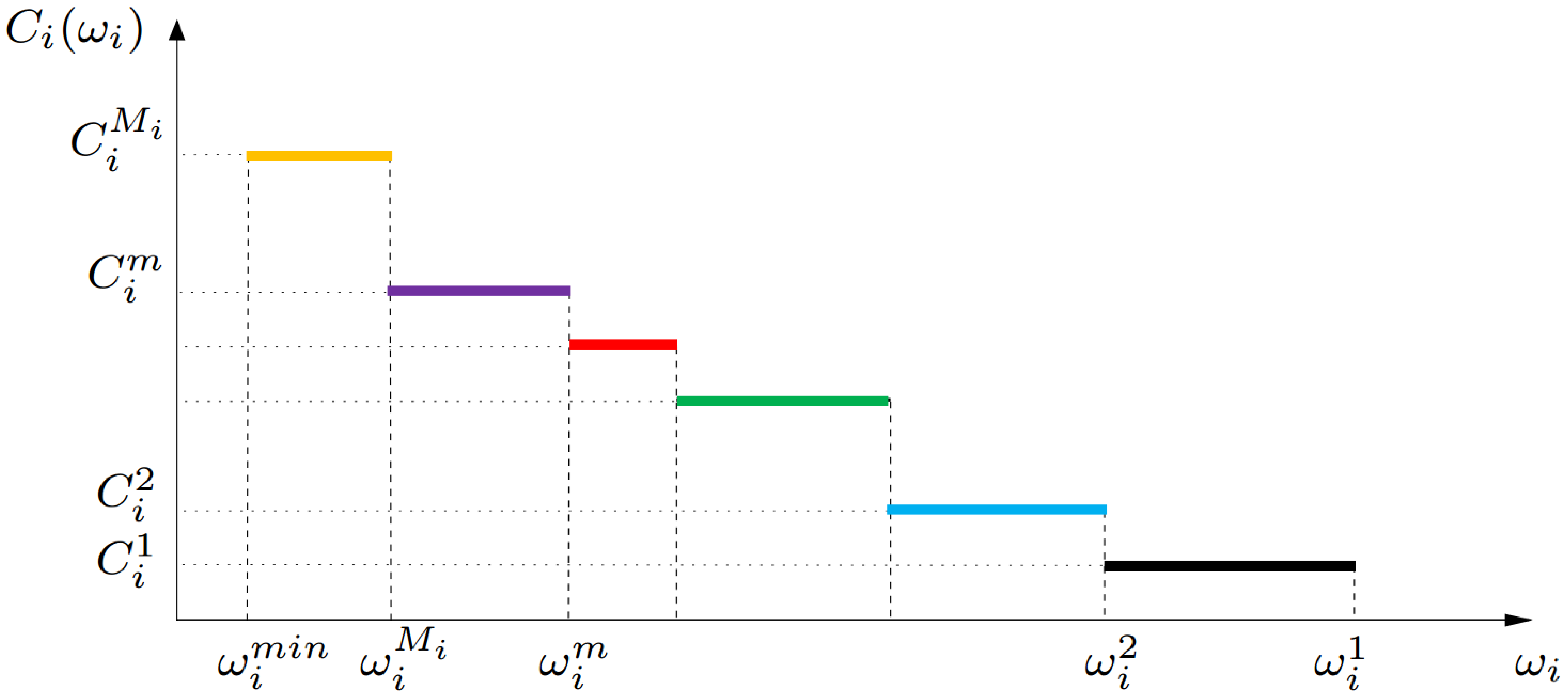
```
#define omega1 1000
#define omega2 2000
#define omega3 4000
#define omega4 60

task    samp

        omeg

        f0();
        if (on                    );
        if (om               f2();
        if (ome          ega2) f3();
        if (omega  ≤ omega1) f4();
}
```

ADAPTIVE
VARIABLE-RATE
TASKS

daptive
ehavior as a
function of the
instantaneous
engine speed

Adaptive Variable-Rate Tasks

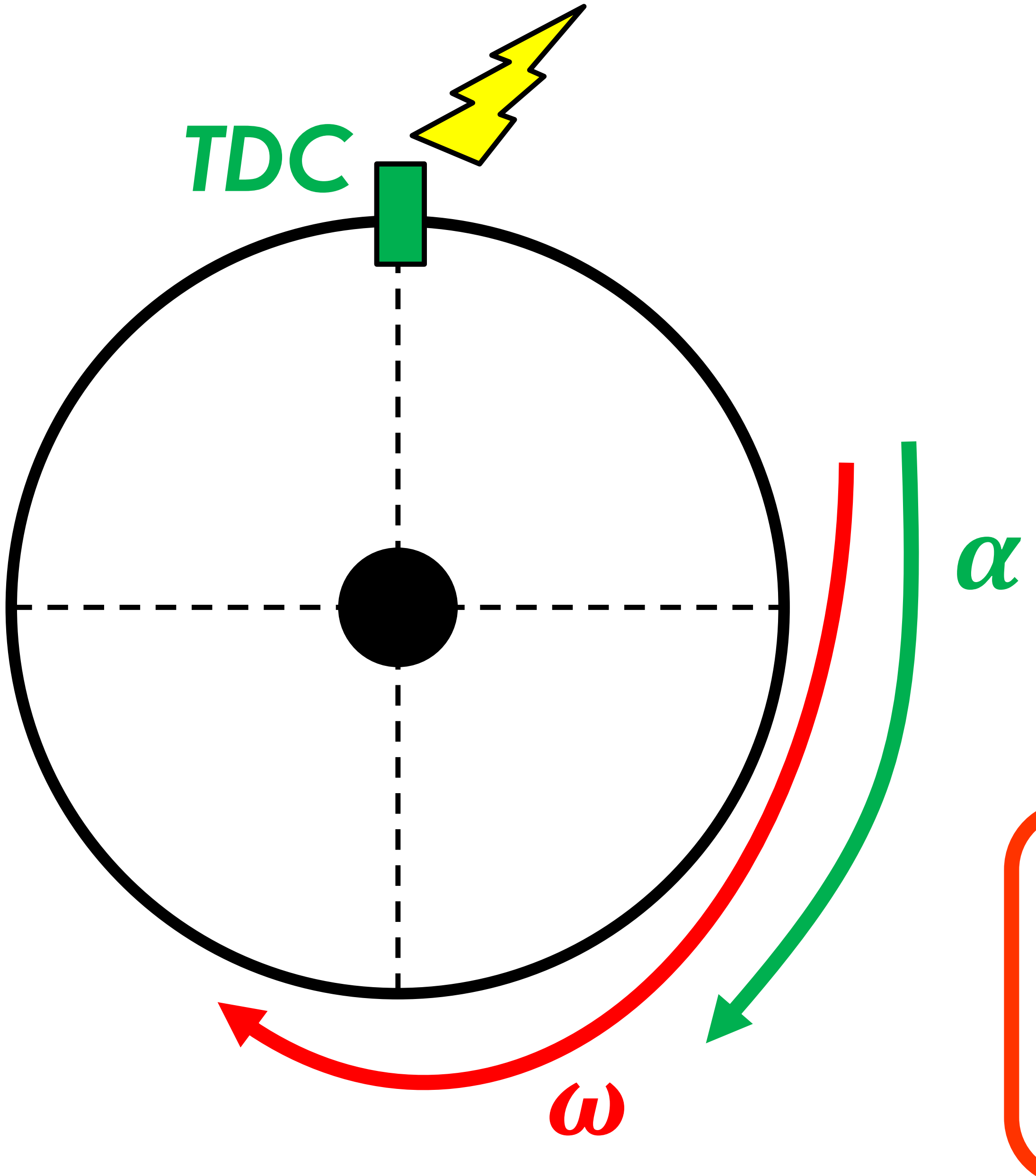
- The AVR task implements a number of **execution modes**



Real-Time Systems Laboratory

AVR Tasks: Dynamic condition

- **Engine-triggered** tasks – **Dynamic** condition



$\alpha > 0, \ \alpha < 0$

Acceleration/Deceleration on the engine speed

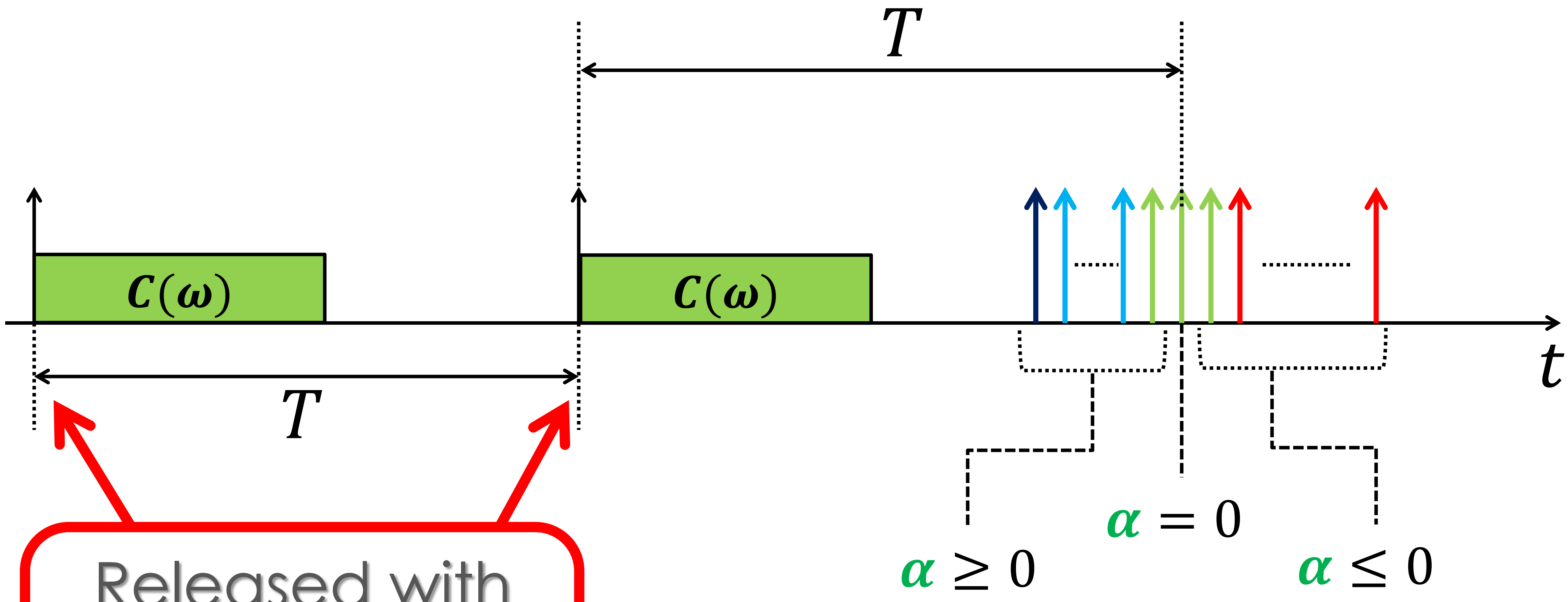
Our model: constant acceleration in a revolution

$\alpha \in [\alpha_{min}, \alpha_{max}]$

Real-Time Systems Laboratory

AVR Tasks: Dynamic condition

- Acceleration $\boldsymbol{\alpha} \epsilon [\alpha_{max}; \alpha_{min}]$, with $\alpha_{min} \leq 0$



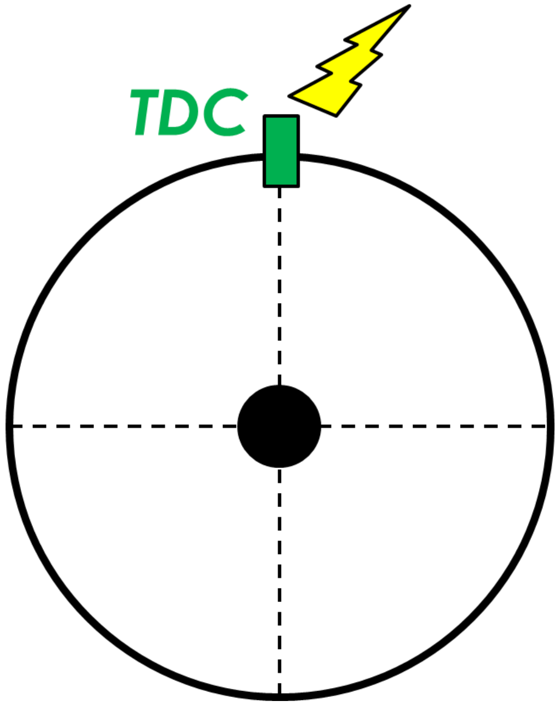
Released with instantaneous engine speed ω

Related Work

- **Kim**, **Lakshmanan**, and **Rajkumar** @ **ICCPS 2012**
  Preliminary work on a simplified model
- **Pollex** *et al.* @ **DATE 2013**
  Sufficient analysis with constant speed
- **Buttazzo**, **Bini** and **Buttle** @ **DATE 2014**
  Analysis in dynamic condition under EDF
- **Davis** *et al.* @ **RTAS 2014**
  Sufficient analysis in dynamic condition under FP using ILP programming and quantization on the speed domain

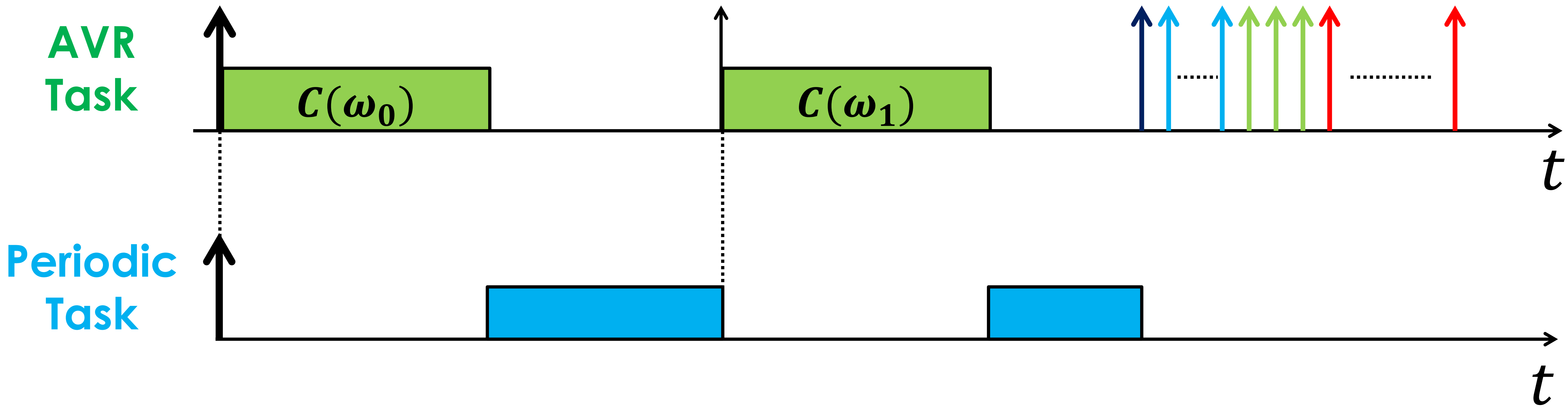
Our work

- ☐ Concentrate on a single **AVR Task** release at TDC (one trigger per revolution);



- ☐ **We studied the problem of deriving the exact worst-case interference of an AVR Task**
  - ☐ Characterize the worst-case computational request in function of the ***engine dynamics*** (i.e., evolution of the speed by accelerations/decelerations).

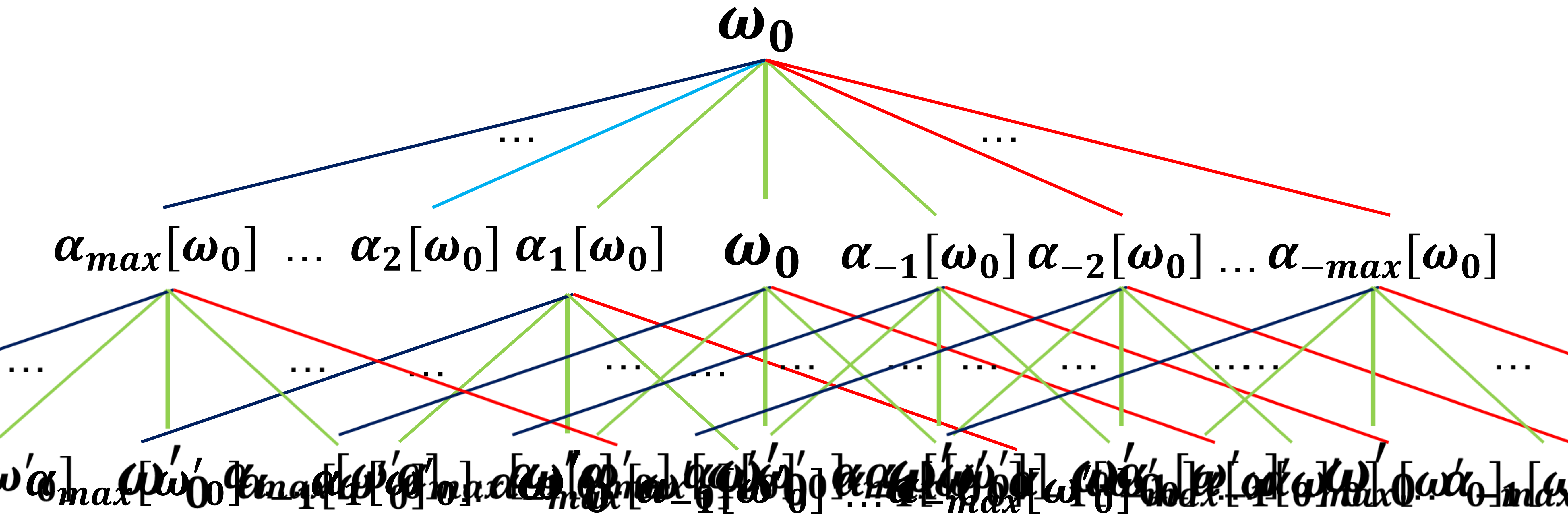
Critical Instant



- Potentially infinite critical instants: one for each instantaneous engine speed ω_0 at which occurs;

- The interference depends on the engine dynamic starting from ω_0.

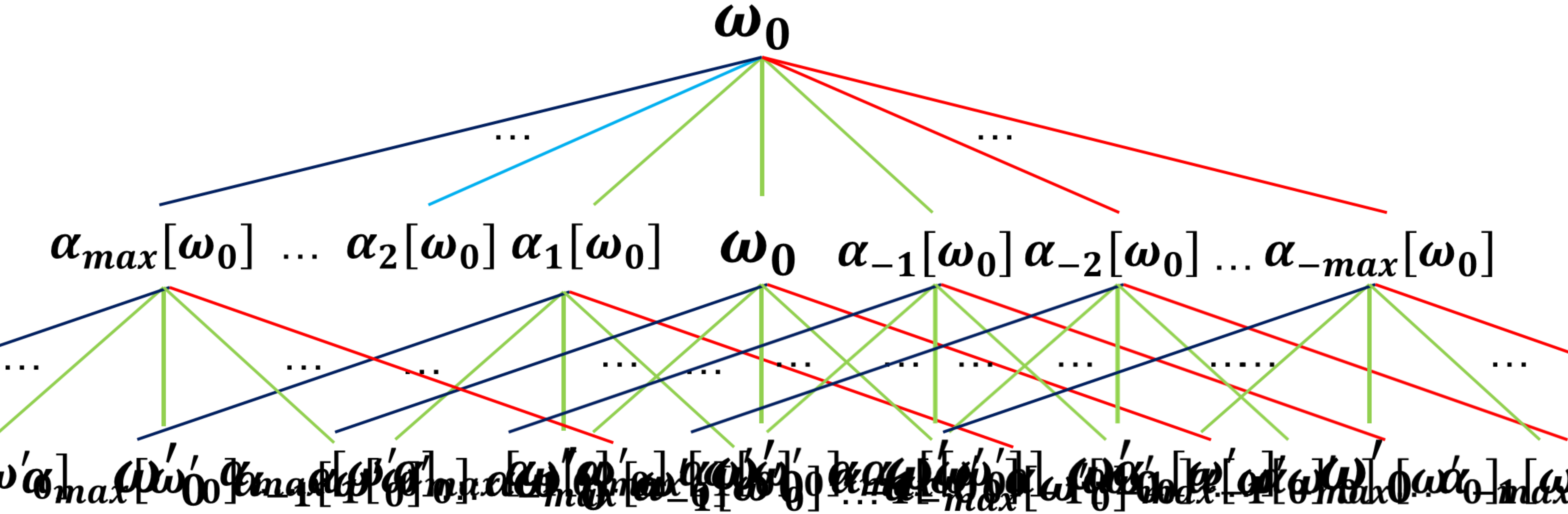
Job Releases



AND SO ON…

…until the end of the interference time window

Job Releases



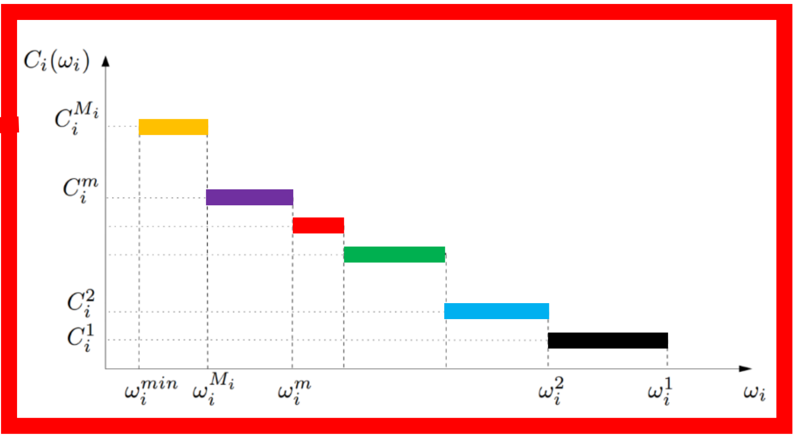
We are interested in the maximum interference of all this possible jobs

Brute-Force Approach

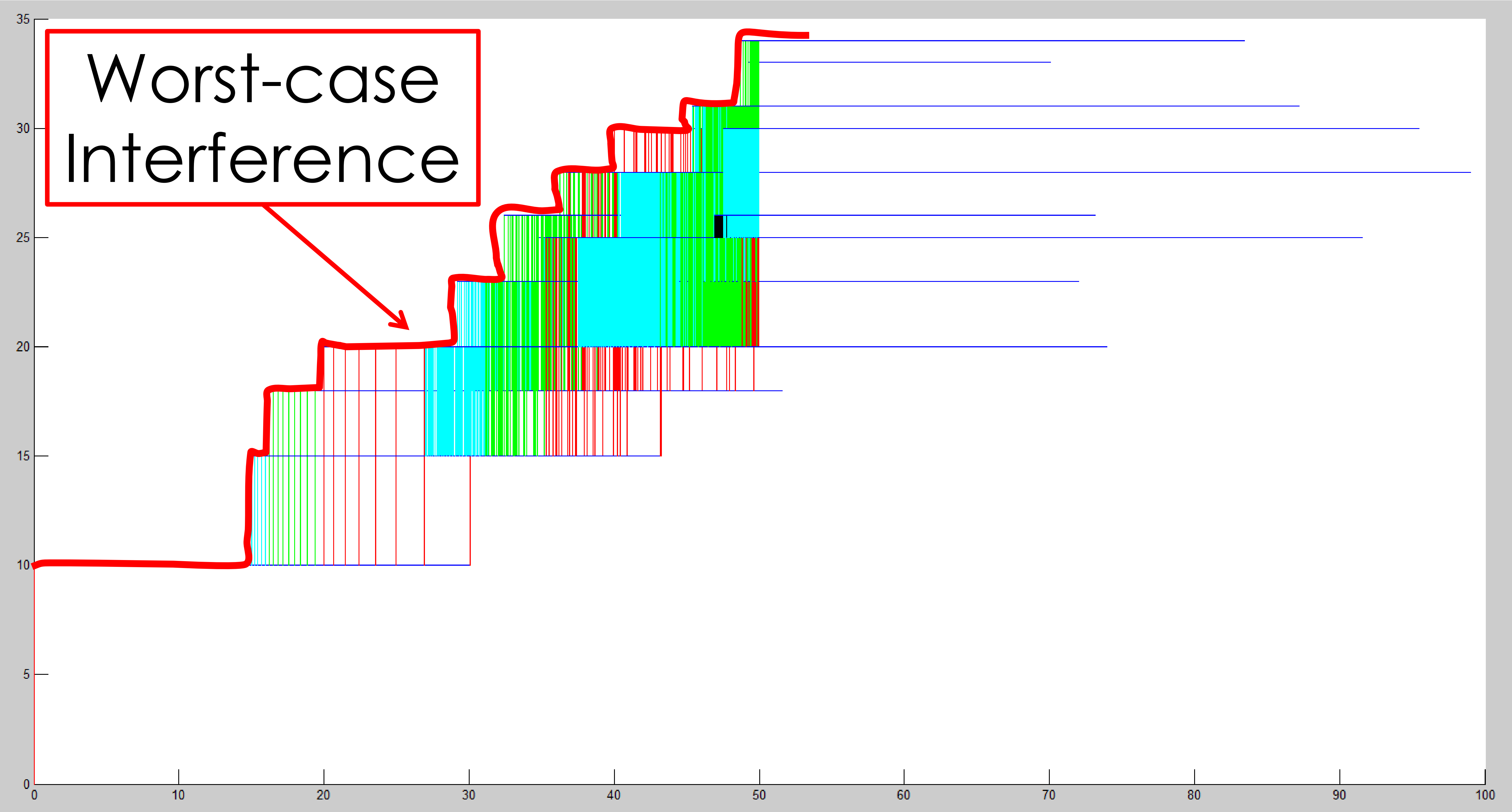
```
Interference($\omega_0$, C, time) {
    if(time>MAX_TIME) return;
    UPDATE_INTERFERENCE (C,time);
    for each $\alpha_{min} \leq \alpha \leq \alpha_{max}$ $step\ \Delta\alpha$ {
        $\omega^{next}$ = $\boldsymbol{\Omega}(\omega, \alpha)$;
        $T^{next}$ = $\boldsymbol{T}(\omega, \alpha)$;
        $C^{next}$ = C + $\boldsymbol{C}(\omega)$;
        Interference ($\omega^{next}$, $C^{next}$,time + $T^{next}$);
    }
}
```

Quantization

Physical equations



Brute-Force Approach



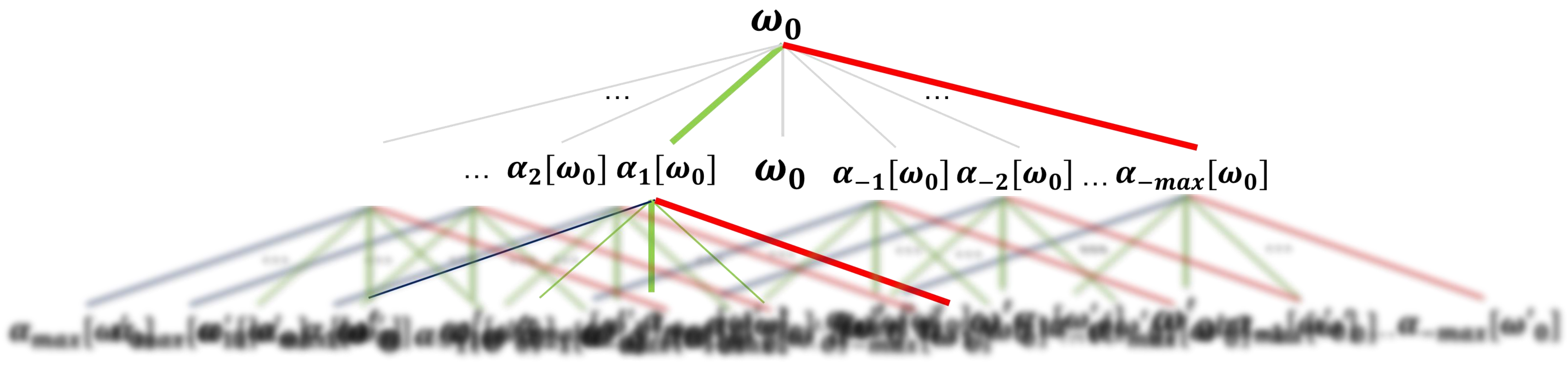
Worst-case Interference

Brute-Force Approach

- `Interference`(ω_0, `C`, `time`)
  - Requires a **complete visit** of the tree;
  - Very expensive in terms of computational complexity, **intractable** for most practical uses;
  - Based on **quantization**.

Pruning

- **Our approach**: derive **pruning rules** to significantly reduce the search complexity;
- We note that ***only a finite set*** of critical job releases must be taken into account to derive the maximum interference.

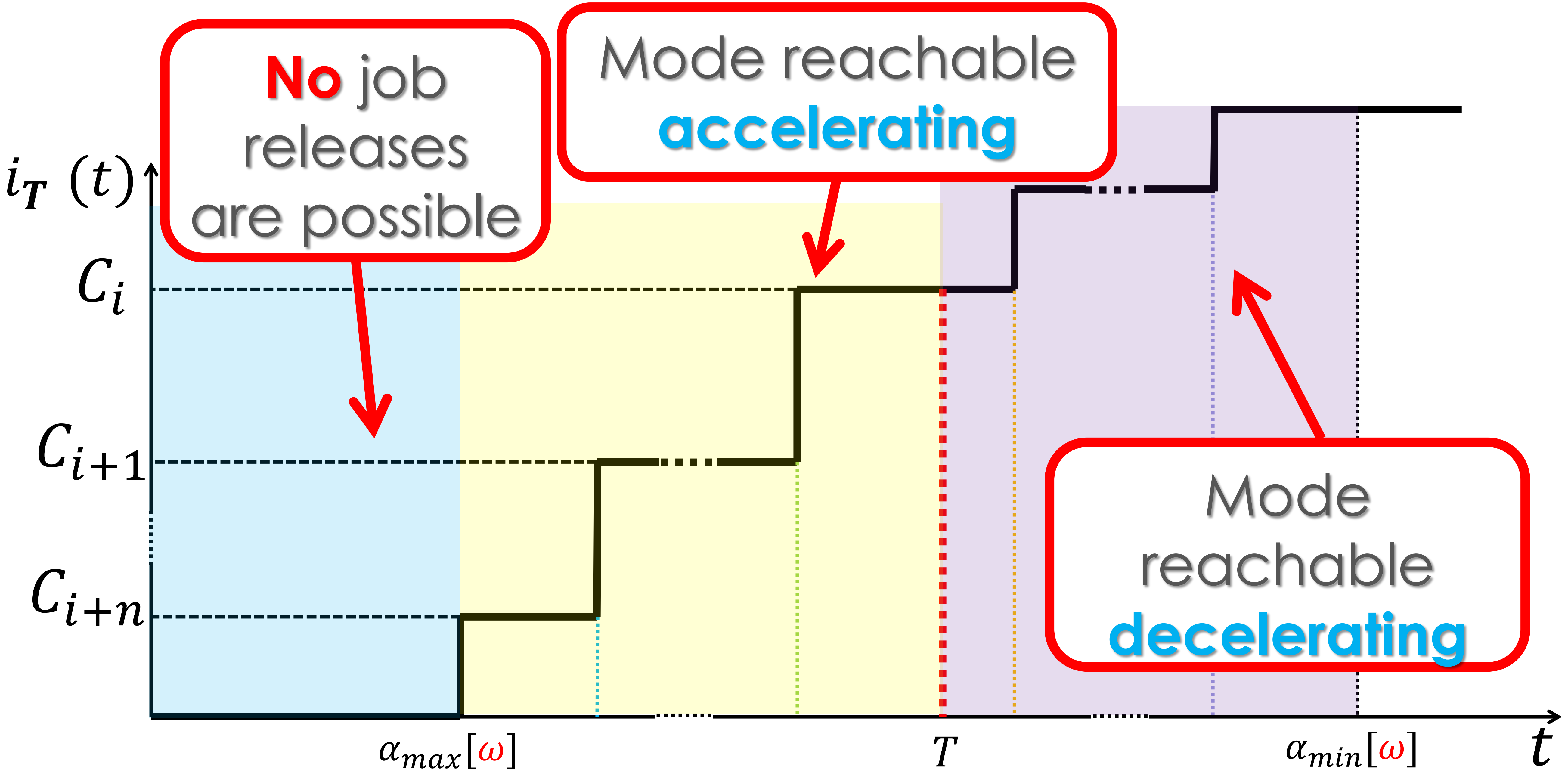


ω_0

… $\alpha_2[\omega_0]$ $\alpha_1[\omega_0]$ ω_0 $\alpha_{-1}[\omega_0]$ $\alpha_{-2}[\omega_0]$ … $\alpha_{-max}[\omega_0]$

Single-Job Interference

- **Interference** of a **single** Job activated with instantaneous speed ω

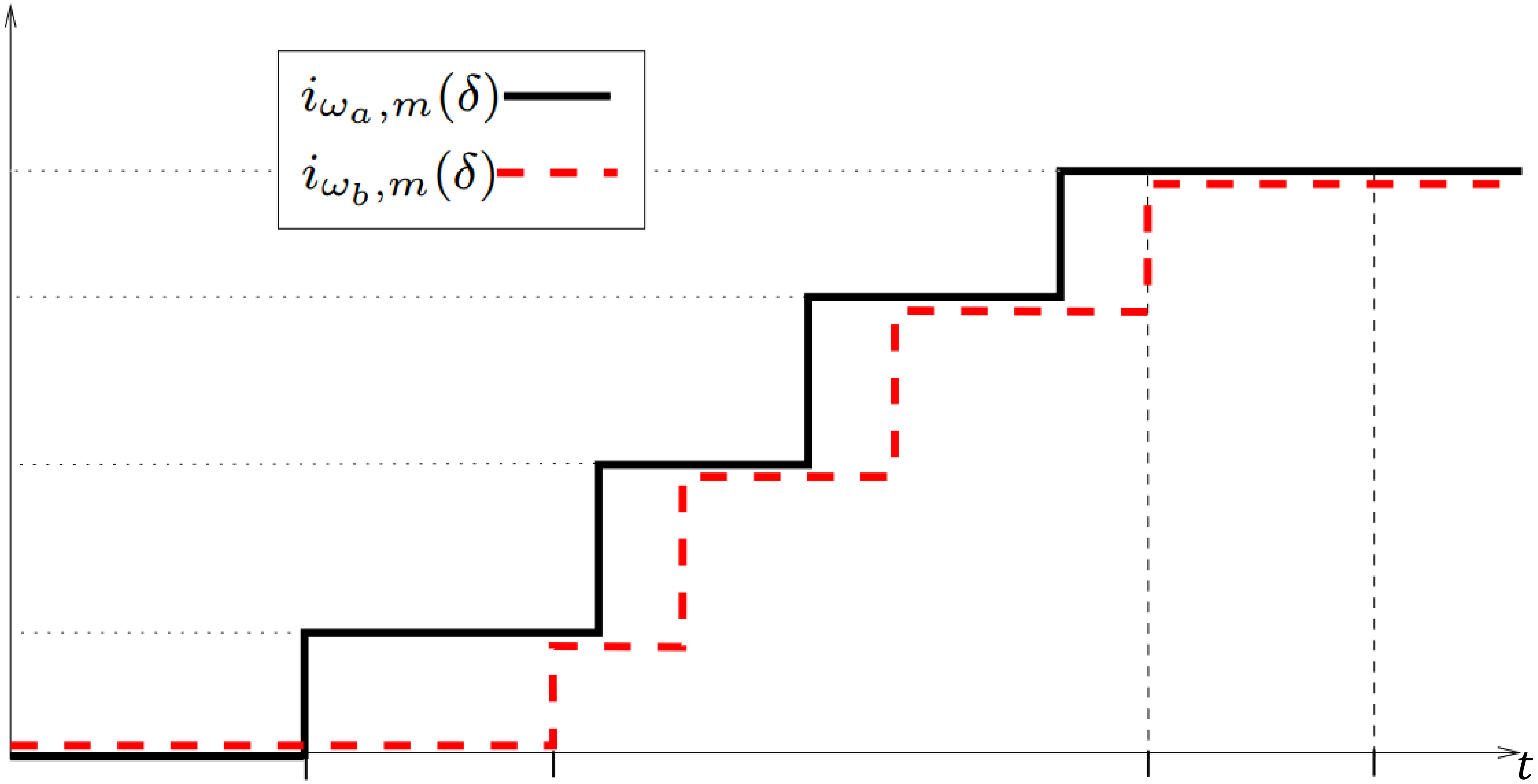


Pruning

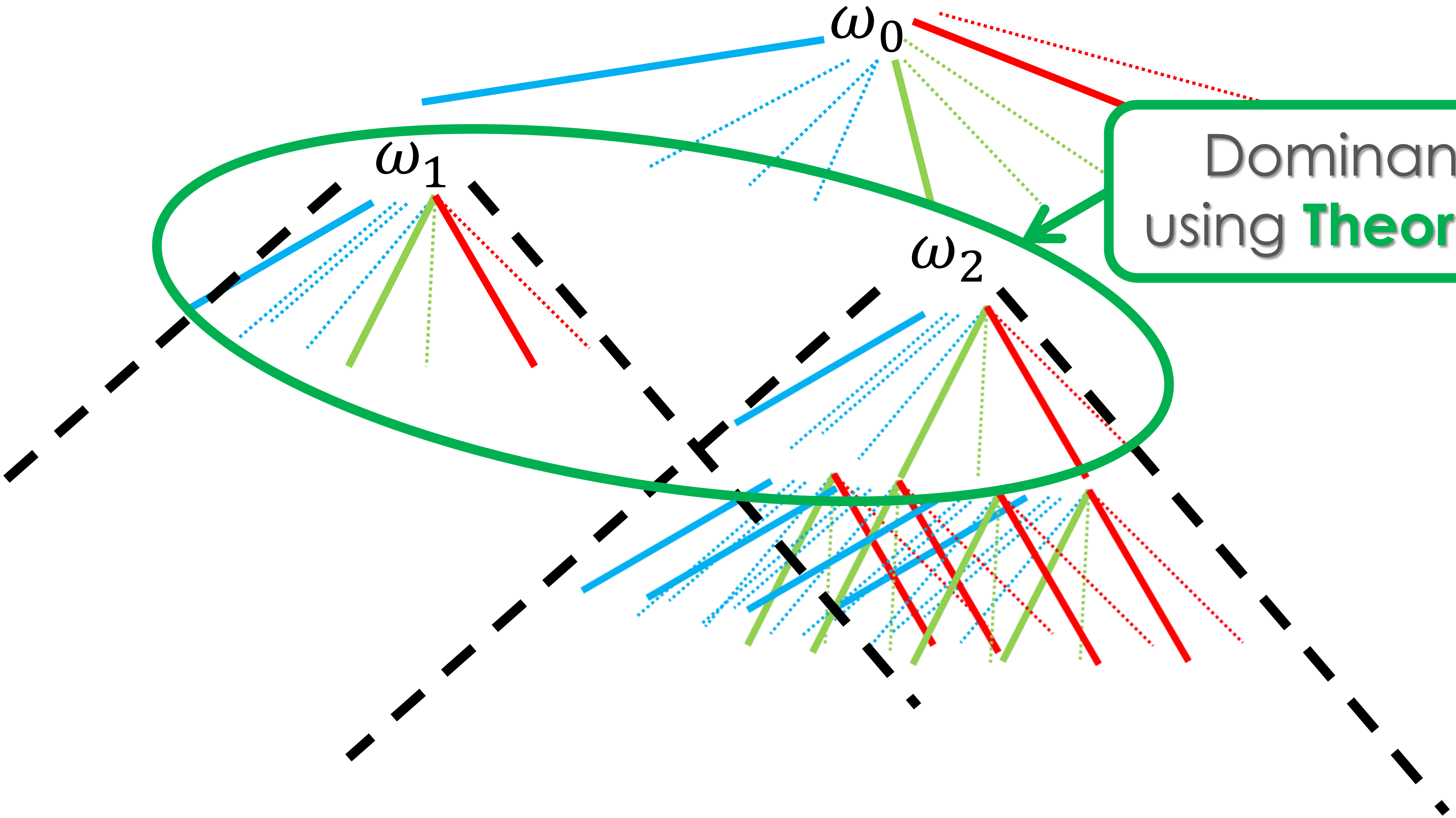
- **Theorem 1-** *dominance on single-job interference*

If $\boldsymbol{\omega_a} \geq \boldsymbol{\omega_b}$ and $\boldsymbol{C}(\boldsymbol{\alpha_{min}}[\boldsymbol{\omega_a}]) = \boldsymbol{C}(\boldsymbol{\alpha_{min}}[\boldsymbol{\omega_b}])$

Then $i_{\omega_a}(t) \geq i_{\omega_b}(t)\ \forall t$

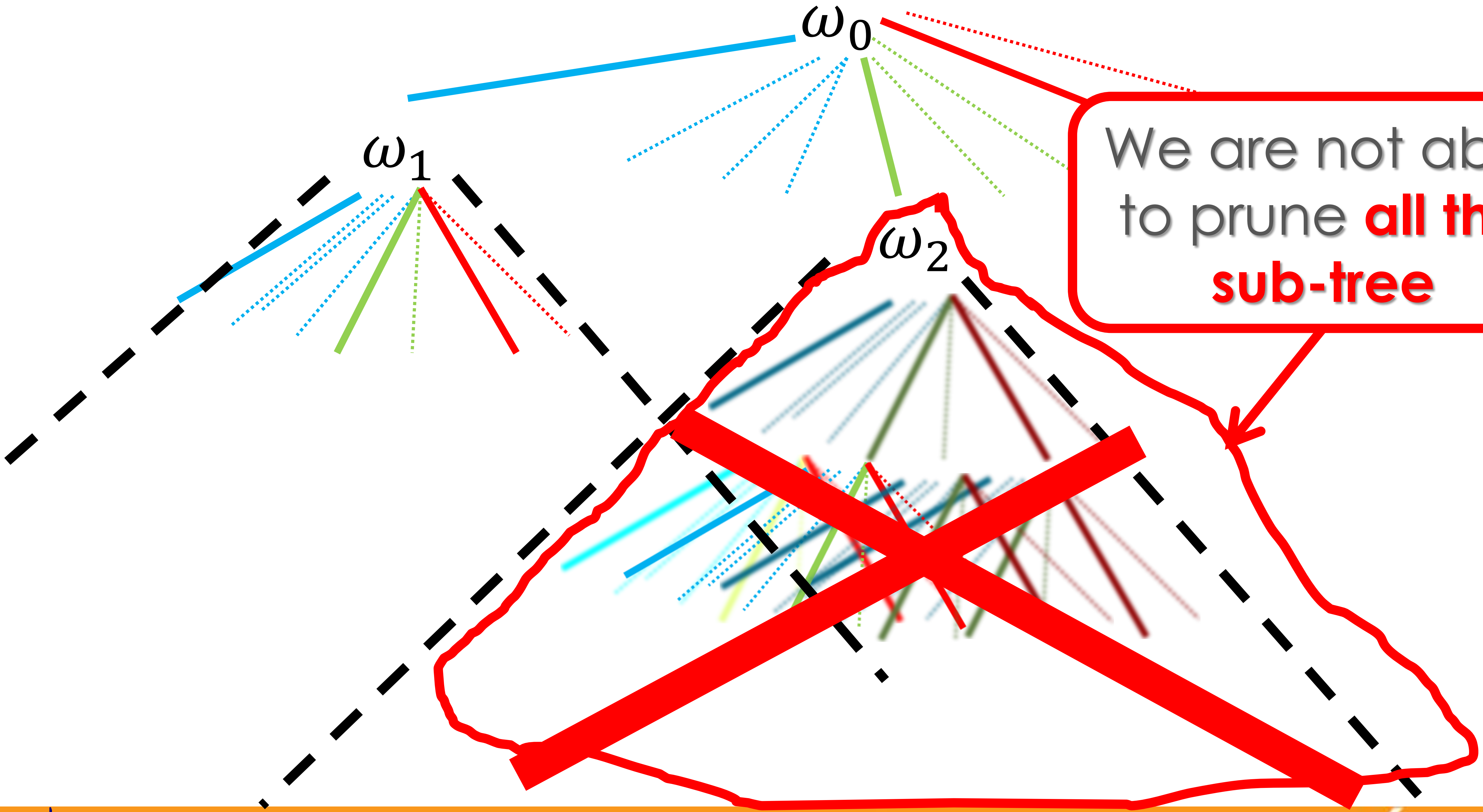


Pruning of Job Sequences



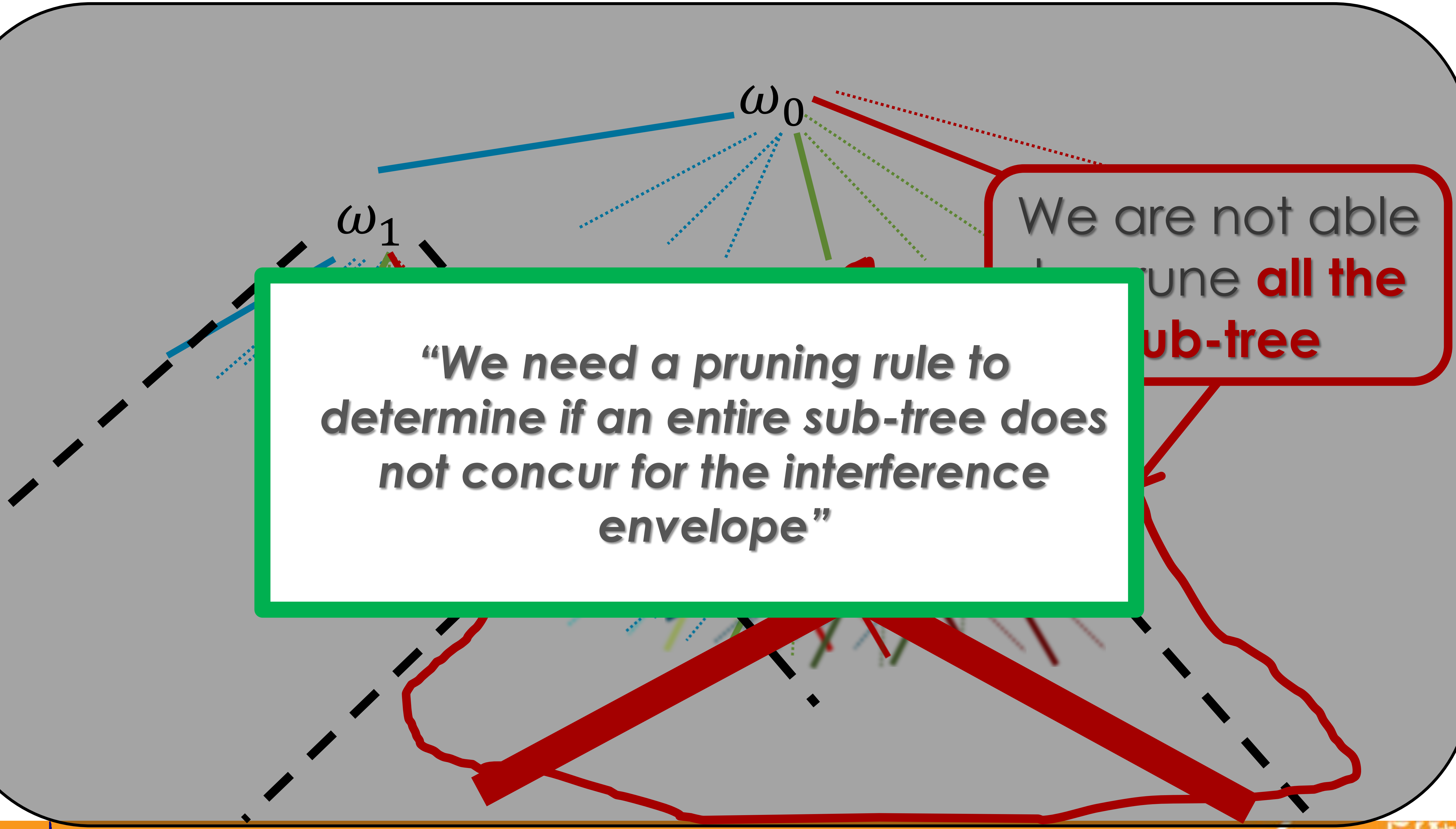
Dominance using **Theorem 1**

Pruning of Job Sequences



We are not able to prune **all the sub-tree**

Pruning of Job Sequences



We are not able

une **all the**

ub-tree

“We need a pruning rule to determine if an entire sub-tree does not concur for the interference envelope”

Pruning

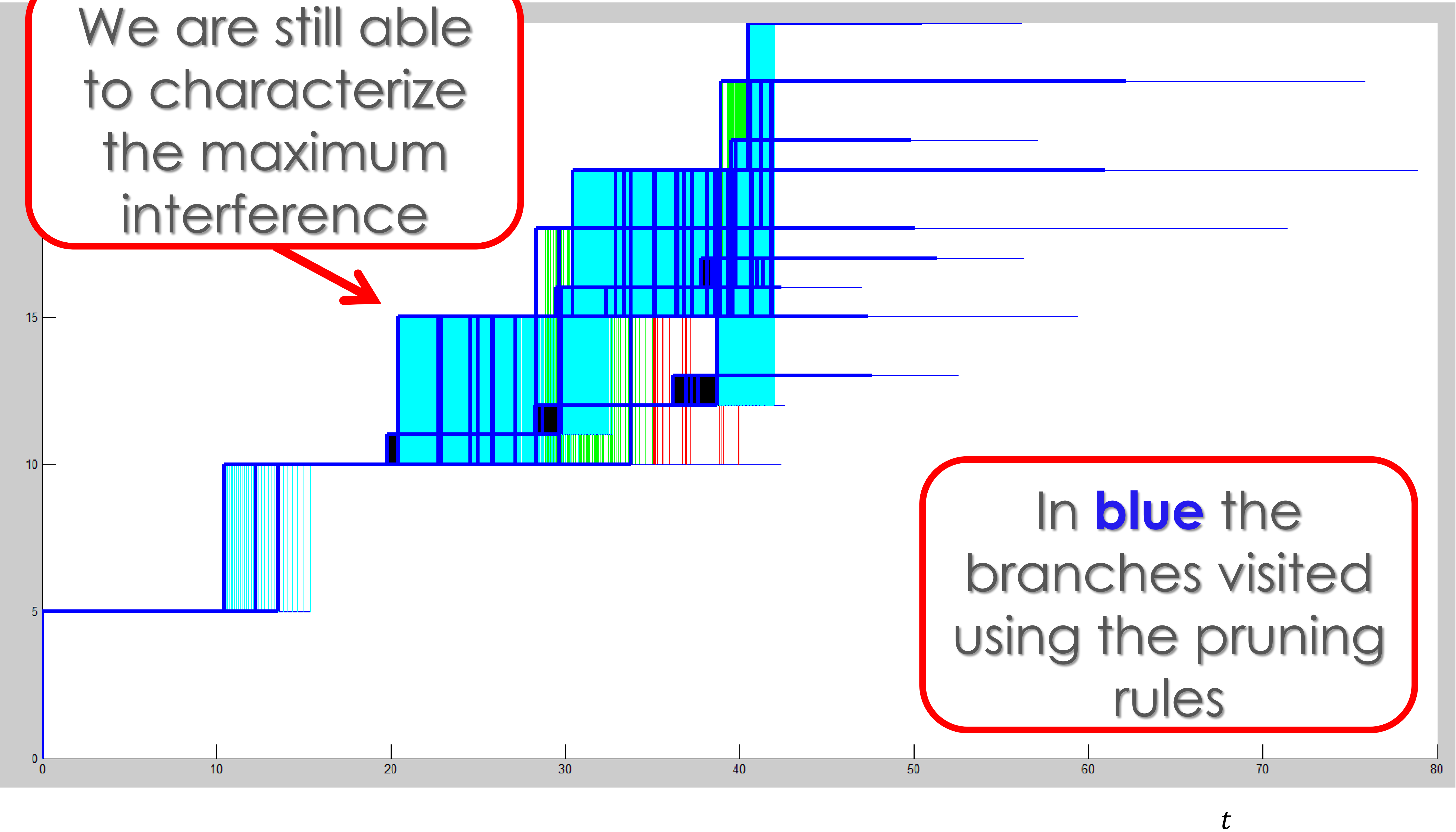
- ❑ **Theorem 2-** *dominance on the sub-tree*

If $\boldsymbol{\omega_a} \geq \boldsymbol{\omega_b}$ and $\boldsymbol{C}(\boldsymbol{\alpha_{min}}[\boldsymbol{\omega_a}]^{\boldsymbol{n}}) = \boldsymbol{C}(\boldsymbol{\alpha_{min}}[\boldsymbol{\omega_b}]^{\boldsymbol{n}})\ \forall \boldsymbol{n} \in \mathbb{N}$

Then $I_{\omega_a}(t) \geq I_{\omega_b}(t)\ \forall t$

It allows to construct an algorithm to prune entire sub-trees, reducing the search domain

Pruning



We are still able to characterize the maximum interference

In **blue** the branches visited using the pruning rules

Pruning

- **Performance –** Compute the interference of an AVR task with 6 modes over a time window of 100ms

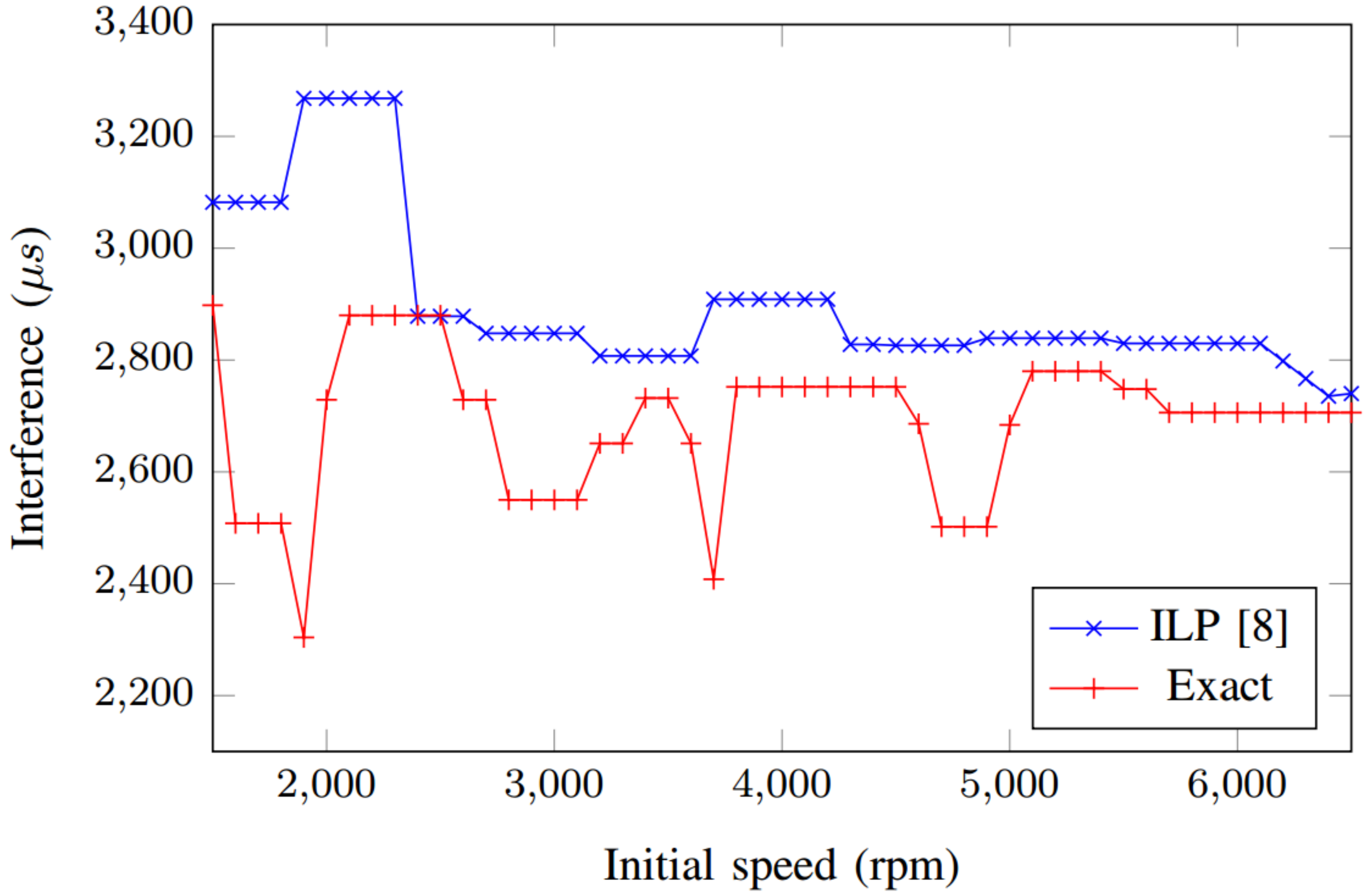
- Implementation as MATLAB scripting
  - ***Brute-force***: ~1 hour;
  - **Pruning-based algorithm**: a few seconds.

Dominant Speeds

- **Recall:** Potentially infinite critical instants: one for each instantaneous engine speed ω_0 at which occurs;
- We have a search tree *for each* initial speed ω_0
- Thanks to **Theorem 2** we are able to identify a limited set of **dominant initial speeds**
  - No quantization;
  - Further improvements in terms of complexity.

Experimental Results

- Comparison with the sufficient **ILP**-based method proposed by ***Davis et al. in RTAS 2014***;
- AVR Task from an application provided in the context of the *INTERESTED EU* project.



Acknowledgements

- Thanks to **Rob Davis**, **Timo Feld**, **Victor Pollex**, and **Frank Slomka** for the interesting and fruitful discussions that helped to improve both this and their work.

Conclusion

- We studied AVR Tasks including engine dynamics;
- We proposed a method to compute an **exact** characterization of the worst-case **interference** of an AVR Task
  - Pruning rules;
  - Dominant initial speeds.

Thank you!

Alessandro Biondi
alessandro.biondi@sssup.it