**Noname manuscript No.**
(will be inserted by the editor)

# Best-case response times and jitter analysis of real-time tasks under fixed priority scheduling with preemption thresholds

**H.J. Rivera-Verduzco · Reinder J. Bril**

Received: date / Accepted: date

**Abstract** Insert your abstract here. Include keywords, PACS and mathematical subject classification numbers as needed.

**Keywords** best-case · response time analysis · preemption thresholds · jitter analysis

## 1 Introduction

Your text comes here. Separate text sections with

## 2 Real-time scheduling model

In this section, we present the basic real-time scheduling model for FPPS and FPTS along with some related notion that we use throughout this paper.

### 2.1 Basic model for FPPS

We assume a single processor and a set $\mathcal{T}$ of $n$ independent periodic tasks $\tau_1, \tau_2, ..., \tau_n$ with unique and fixed priorities $\pi_1$, $\pi_2, ..., \pi_n \in \mathbb{N}^+$. At any moment in time, the processor executes the task with the highest priority that has work pending. We also assume that tasks are given in order of decreasing priority, i.e. $\tau_1$ has the highest priority whereas $\tau_n$ has the lowest priority. A higher priority is represented by a higher value, i.e. $\pi_1 > \pi_2 > ... > \pi_n$.

Each task $\tau_i$ generates an infinite sequence of jobs $\iota_{i,k}$ with $k \in \mathbb{Z}$. The inter arrival times of $\tau_i$ are determined by a (fixed) *period* $T_i \in \mathbb{R}^+$ and an (absolute)

Hector J. Rivera-Verduzco and Reinder J. Bril
Eindhoven University of Technology
E-mail: h.j.rivera.verduzco@student.tue.nl, r.j.bril@tue.nl

*activation jitter* $AJ_i \in \mathbb{R}^+ \cup \{\}$, where $AJ_i < T_i$. Furthermore, each task is characterized by a *worst-case computation time* $WC_i \in \mathbb{R}^+$, a *best-case computation time* $BC_i \in \mathbb{R}^+$, where $BC_i \leq WC_i$, a *phasing* $\phi_i \in \mathbb{R}$, a (relative) *worst-case deadline* $WD_i \in \mathbb{R}^+$, and a (relative) *best-case deadline* $BD_i \in \mathbb{R}^+ \cup \{0\}$, where $BD_i \leq WD_i$. We assume arbitrary deadlines; hence, deadline $WD_i$ may be smaller than, equal to, or larger than period $T_i$. The set of phasings $\phi_i$ is termed the phasing $\phi$ of the task-set $\mathcal{T}$. We also assume that we do not have control over the initial phasing of the system, i.e. arbitrary phasings. Finally, we assume that tasks do not suspend themselves, jobs do not start before the completion of previous jobs of the same task, and the overhead of context switching and task scheduling is ignored.

Throughout this paper, we sometimes use $C_i$ to express the *computation time* of a task $\tau_i$ when $BC_i = WC_i$. In addition, note that the activations of $\tau_i$ do not necessarily take place strictly periodically. Instead, they take place in an interval of length $AJ_i$ that repeats with period $T_i$. In general, the activation time $a_{i,k}$ of a job of $\tau_i$ satisfies

$$\varphi_i + kT_i \leq a_{i,k} \leq \varphi_i + kT_i + AJ_i. \tag{1}$$

A task with activation jitter equal to zero is termed a *strictly periodic* task.

### 2.2 Refined model for FPTS

For FPTS, each task $\tau_i$ has an additional property called *preemption threshold* denoted by $\theta_i \in \mathbb{N}^+$, where $\pi_1 \geq \theta_i \geq \pi_i$. A task $\tau_i$ can be preempted by a higher priority task $\tau_h$ if and only if $\pi_h > \theta_i$. FPPS is a special case of FPTS when the preemption threshold of each task is equal to its priority, i.e. when $\forall_{1 \leq i \leq n} \theta_i = \pi_i$. Furthermore, when all the preemption thresholds are set to the highest priority, FPTS becomes equivalent to FPNS, i.e. when $\forall_{1 \leq i \leq n} \theta_i = \pi_1$.

### 2.3 Derived notions

The *response time* $R_{i,k}$ of a job $\iota_{i,k}$ is defined as the time elapsed between its (absolute) *finalization time* $f_{i,k}$ and (absolute) *activation time* $a_{i,k}$, i.e. $R_{i,k} = f_{i,k} - a_{i,k}$. The *relative finalization time* $F_{i,k}$ of a job $\iota_{i,k}$ is defined relative to the start of the interval in which $\iota_{i,k}$ is activated, i.e. $F_{i,k} = f_{i,k} - (\varphi_i + kT_i)$. The (absolute) *start time* $s_{i,k}$ is the time at which job $k$ of $\tau_i$ starts its execution. Furthermore, the *relative start time* $S_{i,k}$ is the start time relative to the activation of a job, i.e. $S_{i,k} = s_{i,k} - a_{i,k}$. Finally, we define the *hold time* $H_{i,k}$ as the time elapsed between the start of a job and its finalization, this is expressed as $H_{i,k} = f_{i,k} - s_{i,k}$. In general, under this model, it always holds that $R_{i,k} = S_{i,k} + H_{i,k}$. Figure 1 shows the basic task model with these notions. It is worth noting that, whereas $F_{i,k} = R_{i,k}$ holds for a strictly periodic task $\tau_i$, the following relation holds in general

$$AJ_i + R_{i,k} \geq F_{i,k} \geq R_{i,k}. \tag{2}$$

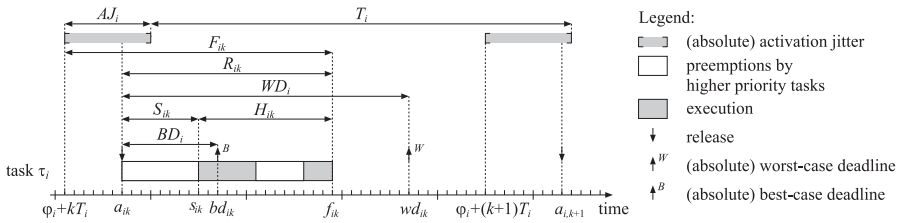


**Fig. 1** Basic model for a periodic task $\tau_i$ with (absolute) activation jitter $AJ_i$.

The *worst-case response time* $WR_i$ and the *best-case response time* $BR_i$ of a task $\tau_i$ are defined as the longest and the shortest response times of its jobs respectively, i.e.

$$WR_i \stackrel{\text{def}}{=} \sup_{\phi,k} R_{i,k}(\phi), \qquad BR_i \stackrel{\text{def}}{=} \inf_{\phi,k} R_{i,k}(\phi),$$

where $R_{i,k}(\phi)$ denotes a dependency of the response time $R_{i,k}$ on the phasing $\phi$.

As is common for critical real-time applications, we assume that deadlines are hard, i.e. each job of a task must be completed at or after its best-case deadline and at or before its worst-case deadline. Therefore, a set $\mathcal{T}$ of $n$ periodic tasks is said to be schedulable if and only if

$$\bigwedge_{1 \le i \le n} (BD_i \le BR_i \wedge WR_i \le WD_i). \tag{3}$$

Finally, we define the *worst-case utilization* $U^{\mathcal{T}}$ and the *best-case utilization* $BU^{\mathcal{T}}$ as the worst and best fraction of the processor time spent on the execution of $\mathcal{T}$ respectively [**?**], i.e.

$$U^{\mathcal{T}} \stackrel{\text{def}}{=} \sum_{1 \le i \le n} \frac{WC_i}{T_i}, \qquad BU^{\mathcal{T}} \stackrel{\text{def}}{=} \sum_{1 \le i \le n} \frac{BC_i}{T_i}.$$

## 3 Recap of the existing analysis

3.1 Best-case response time analysis for FPPS

3.2 Best-case response time analysis for FPTS without jitter??

Add here facts and bcrt analysis of RTNS?

## 4 Towards an optimal instant

Recall that an optimal instant is the time instant that leads to the best-case response time. The best-case response time analysis for FPPS is based on such an optimal instant that occurs when the finalization of a job of a task coincides with the simultaneous activation of all its higher priority tasks. In this chapter, we derive some properties for an optimal instant of a task scheduled

using FPTS. To this end, we first introduce some facts regarding the best-case behaviour of a task scheduled using FPTS. Afterwards, we formally identify the tasks that influence on the best-case response time of another task and we prove some properties for an optimal instant.

### 4.1 Introductory examples

UPDATE THIS WITH NEW EXAMPLES CONSIDERING JITTER.

**Fact 1** *In FPTS, the best-case response time of a task $\tau_i$ can also be affected by a higher priority task that cannot preempt $\tau_i$, i.e. by delaying tasks in $\mathcal{D}_i = \{\tau_d \in \mathcal{T} | \theta_i \geq \pi_d > \pi_i\}$.*

**Fact 2** *In FPTS, the best-case response time of a task $\tau_i$ is not necessarily found when the simultaneous activation of all higher priority tasks that can preempt task $\tau_i$ coincides with the completion of one of its jobs.*

**Fact 3** *The best-case response time under FPTS is not necessarily assumed by the job of $\tau_i$ with the shortest hold time.*

**Fact 4** *Given a set of extra preempting tasks $\mathcal{E}_i$ of task $\tau_i$ and a job $\iota_{i,k}$ experiencing the properties for an optimal instant, let $\boldsymbol{\eta}' \in \mathcal{H}_i(\mathcal{E}_i)$ be the vector of preemptions that leads to the shortest hold time for $\iota_{i,k}$, i.e. $h_i(\boldsymbol{\eta}') = \min_{\boldsymbol{\eta} \in \mathcal{H}_i(\mathcal{E}_i)} \{h_i(\boldsymbol{\eta})\}$. The shortest response time of $\iota_{i,k}$ is not necessarily found when such a job has a vector of preemptions $\boldsymbol{\eta}'$. Therefore, it may hold that $R_{i,k}(\boldsymbol{\eta}) < R_{i,k}(\boldsymbol{\eta}')$ for $\boldsymbol{\eta} \in \mathcal{H}_i(\mathcal{E}_i)$ and $\boldsymbol{\eta} \neq \boldsymbol{\eta}'$.*

OPTIONAL:

**Fact 5** *In FPTS, the best-case response time of a task $\tau_i$ is not necessarily assumed by the last job in a level-i active period.*

### 4.2 Tasks influencing the best-case response time

According to Theorem **??**, lower priority tasks have no influence on the best-case response time of a task $\tau_i$. Therefore, we ignore lower priority tasks in the rest of this document. In addition, we distinguish between two types of tasks that can influence the best-case response time of a task $\tau_i$. These types of tasks are the set $\mathcal{P}_i$ of *preempting* tasks in $\{\tau_p \in \mathcal{T} | \pi_p > \theta_i\}$, and the set $\mathcal{D}_i$ of *delaying* tasks given by Fact 1. We termed the latter type as delaying tasks because they cannot preempt $\tau_i$ but at most can delay its start time. Furthermore, Fact 3 shows that a job $\iota_{i,k^{\text{br}}}$ of task $\tau_i$ that assumes the best-case response time does not necessarily have the shortest hold time. Instead, some preempting tasks may give rise to more preemptions in $\iota_{i,k^{\text{br}}}$. Note that this is the case for the first job of $\tau_4$ starting at time $t = 0$ in Figure **??**. This job assumes the best-case response time; however, it experiences a preemption from $\tau_2$. Based on

this observation, we divide the set $\mathcal{P}_i$ of preempting tasks of $\tau_i$ into the set of preempting tasks that give rise to *extra preemptions* denoted as $\mathcal{E}_i \subseteq \mathcal{P}_i$, and the remaining preempting tasks that we call *minimal preempting* tasks in the set $\mathcal{M}_i = \mathcal{P}_i \setminus \mathcal{E}_i$.

### 4.3 Properties for an optimal instant

We now formulate some properties for an optimal instant for FPTS based on the tasks influencing the best-case response time of a task $\tau_i$.

**Theorem 1** *In FPTS, an optimal instant for a task $\tau_i$, where the best-case response time $BR_i$ is assumed by a job $\iota_{i,k^{\text{br}}}$, has the following properties:*

- ***Activation of higher priority tasks.*** *An optimal instant occurs when the completion of $\iota_{i,k^{\text{br}}}$ coincides with a simultaneous activation of all its minimal preempting tasks in $\mathcal{M}_i$. Furthermore, the activation of all delaying tasks in $\mathcal{D}_i$ and all extra preempting tasks in $\mathcal{E}_i$ coincides with $s_{i,k^{\text{br}}} + \Delta$, where $\Delta$ is a sufficiently small amount of time.*
- ***Activation delays for minimal preempting and delaying tasks.*** *All minimal preempting and delaying tasks experience a maximal activation delay at their respective simultaneous activation, and a minimal activation delay at previous activations.*
- ***Activation delays for extra preempting tasks.*** *Given that an extra preempting task $\tau_e$ preempts $\eta_e$ times to job $\iota_{i,k^{\text{br}}}$, task $\tau_e$ experiences a maximal activation delay in the range $[0, AJ_e]$ at the moment of its simultaneous activation preserving the number of preemptions $\eta_e$. Furthermore, activations before $s_{i,k^{\text{br}}}$ and in the interval $(s_{i,k^{\text{br}}} + \Delta, f_{i,k^{\text{br}}})$ experience a minimal activation delay.*

*Proof*

Note that, given a task-set $\mathcal{T}$ and a task $\tau_i \in \mathcal{T}$, the set of delaying tasks $\mathcal{D}_i$ is empty when the preemption threshold of $\tau_i$ is equal to its priority. Furthermore, the set of extra preempting tasks $\mathcal{E}_i$ is always empty when there are no delaying tasks. Therefore, we conclude that the properties of an optimal instant for a task $\tau_i$ scheduled under FPTS described in Theorem 1 gives rise to an optimal instant for FPPS when $\theta_i = \pi_i$.

Figure **??**[1] shows an example of these properties for an optimal instant under FPTS for task $\tau_i$. As can be seen, the activation of the minimal preempting task $\tau_{p_1}$ coincides with the completion of the job of task $\tau_i$ at time $t = 0$. Furthermore, delaying task $\tau_d$ and extra preempting task $\tau_{p_2}$ are activated an instant $\Delta$ after the start of $\tau_i$.

It is worth noting that, from the properties for an optimal instant given above, it is not clear how to partition the set of preempting tasks $\mathcal{P}_i$ into the (sub-)sets of extra preempting $\mathcal{E}_i$ and minimal preempting $\mathcal{M}_i$ tasks. In fact, we

[1] PENDING TO ADD FIGURE

could have chosen $\mathcal{E}_i = \{\tau_{p_1}\}$ and $\mathcal{M}_i = \{\tau_{p_2}\}$ in the example depicted in Figure **??**, and we would still have obtained an option that is in accordance with the properties described in Theorem 1. However, this new option does not lead to the best-case response time. Therefore, we conclude that in FPTS there may be multiple candidates of optimal instants that we have to explore in order to find the best-case response time of a task $\tau_i$. Clearly, this differs from the best-case analysis for FPPS, where the way to construct an optimal instant is always unique. Given this observation, our best-case analysis is based on exploring all possible partitions of preempting tasks into the sets of extra preempting and minimal preempting tasks.

### 4.4 The activation delay of higher priority tasks

Recall from Theorem 1 that the activation delay $\alpha_j$ of a minimal preempting or a delaying task $\tau_j$ is maximal at the moment of its simultaneous activation; hence, it holds that

$$\alpha_j = AJ_j \qquad \text{for all } \tau_j \in \mathcal{M}_i \cup \mathcal{D}_i. \tag{4}$$

Furthermore, the maximal activation delay of an extra preempting task $\tau_e$ at the moment of its simultaneous activation is constrained by the number of times $\eta_e$ that preempts to the job with a best-case response time; moreover, this activation delay is in the range $[0, AJ_e]$. Therefore, it is necessary to first determine the number of preemptions for each of the extra preempting tasks to derive their activation delays. To this end, we first introduce the notion of *preemption vector* as follows.

**Definition 1** *Preemption vector*. Let $\iota_{i,k^{\text{opt}}}$ be a job of $\tau_i$ that experiences the properties for an optimal instant as described in Theorem 1. A preemption vector $\boldsymbol{\eta}$ of $\iota_{i,k^{\text{opt}}}$ is a $|\mathcal{P}_i|$-dimensional vector such that $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_{|\mathcal{P}_i|})$ where $\eta_p$ corresponds to the number of times that a preempting task $\tau_p$ preempts to $\iota_{i,k^{\text{opt}}}$.

Based on a preemption vector, the calculation of the hold time of a job experiencing the properties for an optimal instant is straightforward.

**Corollary 1** *Let $\iota_{i,k^{\text{opt}}}$ be a job of $\tau_i$ that experiences the properties for an optimal instant and let $\boldsymbol{\eta}$ be its corresponding preemption vector. The hold time $h_i(\boldsymbol{\eta})$ of $\iota_{i,k^{\text{opt}}}$ is given by*

$$h_i(\boldsymbol{\eta}) = BC_i + \sum_{p:\tau_p \in \mathcal{P}_i} \eta_p \cdot BC_p. \tag{5}$$

Figure 2 shows and example of the activation delay $\alpha_e$ for an arbitrary extra preempting task $\tau_e$. Note that the first job of $\tau_e$ is activated in the middle of its activation interval at a time $t = s_{i,k^{\text{opt}}} + \Delta$ resulting in $\alpha_e < AJ_e$. Furthermore, note that $\alpha_e$ cannot increase because $\eta_e = 3$. Increasing $\alpha_e$ would result in an

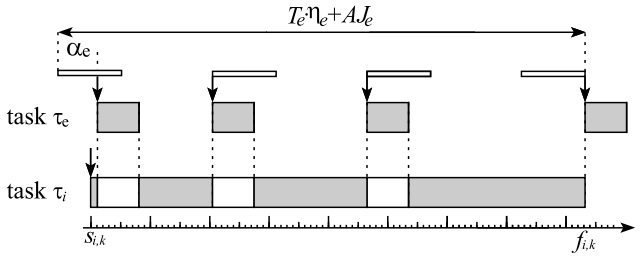


**Fig. 2** An example where the activation delay $\alpha_e$ of an extra preempting task $\tau_e$ is constrained by the number of times $\eta_e = 3$ that it preempts to job $\iota_{i,k^{\text{opt}}}$ as described in Theorem 1. Furthermore, note that the activation delay of the jobs of $\tau_e$ during interval $(s_{i,k^{\text{opt}}} + \Delta, f_{i,k^{\text{opt}}})$ is minimal.

additional preemption by the last job of $\tau_e$ in job $\iota_{i,k^{\text{opt}}}$; hence, violating the constraint of $\eta_e = 3$. We now derive a corollary to determine the maximal activation delay of extra preempting tasks as follow.

**Corollary 2** *Let $\iota_{i,k^{\text{opt}}}$ be a job of $\tau_i$ that experiences the properties for an optimal instant and let $\boldsymbol{\eta}$ be its corresponding preemption vector. Furthermore, let $\tau_e$ be an extra preempting task of such a job. The maximal activation delay $\alpha_e(\boldsymbol{\eta}, \Delta)$ of the job of $\tau_e$ activated at time $t = s_{i,k^{\text{opt}}} + \Delta$ is given by*

$$\alpha_e(\boldsymbol{\eta}, \Delta) = \min(T_e \cdot \eta_e + AJ_e + \Delta - h_i(\boldsymbol{\eta}), AJ_e)^+ \qquad \text{for all } \tau_e \in \mathcal{E}_i. \tag{6}$$

The outer min and superscript (+) operator in Equation (6) keep $\alpha_e(\boldsymbol{\eta}, \Delta)$ within the range $[0, AJ_e]$. Furthermore, the term $T_e \cdot \eta_e + AJ_e$ is the time length from the start of the activation interval of the job of $\tau_e$ activated just after $\iota_{i,k^{\text{opt}}}$ starts and the end of the activation interval of the first job of $\tau_e$ that is activated at $t \geq f_{i,k^{\text{opt}}}$. Figure 2 depicts this situation. Combining Equations (4) and (6), we derive a more general formula for the activation delay of higher priority tasks as follows.

$$\alpha_j(\boldsymbol{\eta}, \Delta) = \begin{cases} \min(T_j \cdot \eta_j + AJ_j + \Delta - h_i(\boldsymbol{\eta}), AJ_j)^+ & \text{for all } \tau_j \in \mathcal{E}_i \\ AJ_j & \text{for all } \tau_j \in \mathcal{M}_i \cup \mathcal{D}_i \end{cases} \tag{7}$$

## 5 Determining the preemption vectors of a job

From Corollary 2, we have to determine first the preemption vector of a job that assumes the best-case response time in order to determine the activation delays of its extra preempting tasks and subsequently construct an optimal instant. Note that this job does not necessarily has a preemption vector that leads to the shortest hold time (Fact 3). Hence, we propose to explore all possible preemption vectors of a job experiencing an optimal instant in order to determine the one that leads to the best-case response time.

In this section, we first show how to determine the largest and shortest preemption vectors of a job given a set of extra preempting tasks. In case these vectors differ, we compute the preemption vectors falling in between.

### 5.1 Upper and lower bounds for the preemption vectors

Clearly, a preemption vector of a job $\iota_{i,k^{\text{opt}}}$ experiencing the properties for an optimal instant is closely related to the hold time $H_{i,k^{\text{opt}}}$ of such a job. When $\iota_{i,k^{\text{opt}}}$ experiences more preemptions (a larger preemption vector), its hold time will increase. Therefore, we propose to bound the hold time that such a job can assume in order to determine its corresponding preemption vectors. Furthermore, recall that the properties for an optimal instant are based on a partition of preempting tasks. Therefore, we found a bound on the hold time given a set of extra preempting tasks. To this end, we first introduce the notion of *upper hold interval* to determine the hold time of a job experiencing a maximal interference by its extra preempting tasks.

**Definition 2** The *upper hold interval* $HI_i^{\text{up}}(y, \mathcal{E}_i)$ is defined as the maximal hold time of a job $\iota_{i,k}$ of an artificial task $\tau_i'$ with a computation time $y \in \mathbb{R}^+$ when such a job $\iota_{i,k}$ experiences extra preemptions by the tasks in $\mathcal{E}_i$, i.e. the tasks in $\mathcal{E}_i$ are activated just after job $\iota_{i,k}$ starts. In addition, all jobs in the interval assume their best-case computation times.

The upper hold interval $HI_i^{\text{up}}(y, \mathcal{E}_i)$ follows directly from the formula for worst-case hold time for FPTS given in [**?**]. The only difference is that best-case computation times are assumed for the maximal hold interval. Hence, $HI_i^{\text{up}}(y, \mathcal{E}_i)$ is given by the smallest $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{e:\tau_e \in \mathcal{E}_i} \left\lceil \frac{x + AJ_e}{T_e} \right\rceil BC_e. \tag{8}$$

Note that the term $+AJ_e$ in the formula guarantees a maximal interference by extra preempting tasks in the hold time. Moreover, Equation (8) can be solved by an iterative procedure starting with a lower bound.

Similar to the notion of upper hold interval, we also introduce the *lower hold interval* to express the hold time of a job when experiencing a minimal interference by its extra preempting tasks. This latter notion will serve as a basis to derive a lower bound on the hold time of a job experiencing an optimal instant.

**Definition 3** The *lower hold interval* $HI_i^{\text{low}}(y, \mathcal{E}_i)$ is defined as the minimal hold time of a job $\iota_{i,k}$ of an artificial task $\tau_i'$ with a computation time $y \in \mathbb{R}^+$ when such a job $\iota_{i,k}$ experiences extra preemptions by the tasks in $\mathcal{E}_i$.

The lower hold interval $HI_i^{\text{low}}(y, \mathcal{E}_i)$ is given by the smallest $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{e:\tau_e \in \mathcal{E}_i} \left\lceil \frac{x - AJ_e}{T_e} \right\rceil^* BC_e, \tag{9}$$

where the notation $w^*$ stands for $\max(w, 1)$. Note that the term $-AJ_e$ in the formula guarantees a minimal interference by extra preempting tasks in the hold time. Furthermore, the superscript $(*)$ is added to guarantee that every extra

preempting task preempts the job under consideration at least once. Equation (9) can be solved by an iterative procedure starting with a lower bound.

So far, we have introduced notions to determine the hold time of a job based on the maximal and minimal interference by extra preempting tasks. It only remains to consider the influence that minimal preempting tasks induce on such a hold time. Observe that, in contrast with extra preempting tasks, the influence of minimal preempting tasks in the hold time of a job experiencing an optimal instant is always minimal. Hence, we introduce the following notion.

**Definition 4** The *minimal hold interval* $HI_i^{\min}(y, \mathcal{E}_i)$ is defined as the minimal hold time of a job $\iota_{i,k}$ of an artificial task $\tau_i'$ with a computation time $y \in \mathbb{R}^+$ when such a job $\iota_{i,k}$ experiences minimal preemptions by the tasks in $\mathcal{M}_i$.

Since $\mathcal{M}_i$ only contains preempting tasks, we can use the same formula for the best-case hold time for FPPS given in [**?**][2] to calculate $HI_i^{\min}(y, \mathcal{M}_i)$. Hence, function $HI_i^{\min}(y, \mathcal{M}_i)$ returns the largest positive solution of $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{m:\tau_m \in \mathcal{M}} \left( \left\lceil \frac{x - AJ_m}{T_m} \right\rceil - 1 \right)^+ BC_m. \tag{10}$$

.

---

Finally, we introduce the set of hold times of a task $\tau_i$ based on its extra and minimal preempting tasks.

**Definition 5** The set of hold times $\mathcal{H}_i(\mathcal{E}_i)$ is defined as the set of all possible hold times that a job $\iota_{i,k}$ of $\tau_i \in \mathcal{T}$ can have when experiencing extra preemptions by the tasks in $\mathcal{E}_i \subseteq \mathcal{P}_i$ and minimal preemptions by the tasks in $\mathcal{M}_i = \mathcal{P}_i \setminus \mathcal{E}_i$. This is, when the tasks in $\mathcal{E}_i$ are simultaneously activated at time $s_{i,k} + \Delta$, where $\Delta$ is a sufficiently small amount of time to experience such extra preemptions. Furthermore, delaying tasks of $\tau_i$ in $\mathcal{T}$ are ignored.

Clearly, given a set of extra preempting tasks $\mathcal{E}_i$, a hold time $h \in \mathcal{H}_i(\mathcal{E}_i)$ is given by the following equation:

$$h = \beta_M + \beta_E + BC_i, \tag{11}$$

where $\beta_M, \beta_E \in \mathbb{R}^+ \cup \{0\}$ are the amount of time spent by preemptions of minimal preempting and extra preempting tasks in the hold time $h$ respectively. Furthermore, it is possible to determine the minimum possible values of $\beta_M$ and $\beta_E$ by solving the following set of recursive equations starting with a lower bound:

$$\begin{aligned} \beta_E^{\text{mi}} &= HI_i^{\text{mi}}(\beta_M^{\text{mi}} + BC_i, \mathcal{E}_i) - \beta_M^{\text{mi}} - BC_i, \\ \beta_M^{\text{mi}} &= BH_i(\beta_E^{\text{mi}} + BC_i, \mathcal{M}_i) - \beta_E^{\text{mi}} - BC_i. \end{aligned} \tag{12}$$

Note that the first equation in (12) minimizes the influence of extra preempting tasks using the minimal hold interval $HI_i^{\text{mi}}(y, \mathcal{E})$. On the other hand, the second

[2] In [**?**], the term best-case hold time is referred as best-case execution time.

equation minimizes the influence of minimal preempting tasks using the best-case hold interval $BH_i(y, \mathcal{M})$. Similarly, the maximum possible values of $\beta_M$ and $\beta_E$ can be derived using the maximal hold interval $HI_i^{\text{ma}}(y, \mathcal{E})$. More precisely, by solving the following set of recursive equations starting with an upper bound. This set of equations maximizes the influence of extra preempting tasks in the hold time.

$$\begin{aligned} \beta_E^{\text{ma}} &= HI_i^{\text{ma}}(\beta_M^{\text{ma}} + BC_i, \mathcal{E}_i) - \beta_M^{\text{ma}} - BC_i, \\ \beta_M^{\text{ma}} &= BH_i(\beta_E^{\text{ma}} + BC_i, \mathcal{M}_i) - \beta_E^{\text{ma}} - BC_i. \end{aligned} \tag{13}$$

Once the maximum and minimum values of $\beta_M$ and $\beta_E$ are determined using the sets of recursive equations in (12) and (13), the minimum ($h^{\text{mi}}$) and the maximum ($h^{\text{ma}}$) hold times of a job in the set $\mathcal{H}_i(\mathcal{E}_i)$ can be derived using Equation (11). In general, the set of hold times $\mathcal{H}_i(\mathcal{E}_i)$ may contain values in the range $[h^{\text{mi}}, h^{\text{ma}}]$. However, when $h^{\text{ma}} = h^{\text{mi}}$, then this is the only possible hold time for a job of $\tau_i$ that experiences extra preemptions by the tasks in $\mathcal{E}_i$; hence, $\mathcal{H}_i(\mathcal{E}_i)$ is a singleton for this case.

As an example where more than one hold time can be found, consider the set of tasks with characteristics as described in Table 1. Let $\tau_1$ be an extra preempting task of $\tau_4$, and $\tau_2$ a minimal preempting task, i.e. $\mathcal{E}_4 = \{\tau_1\}$ and $\mathcal{M}_4 = \{\tau_2\}$. After solving the sets of recursive equations (13) and (12) for task $\tau_4$, we find that the maximum and minimum hold times of a job of $\tau_4$ are $h^{\text{ma}} = 24$ and $h^{\text{mi}} = 18$ respectively. Figure 3 depicts two timelines in which these hold times are found for a job of task $\tau_4$. As can be seen, in both cases $\tau_1$ is activated just after the start time of the job of $\tau_4$; hence, $\tau_1$ is an extra preempting task. In addition, an activation of $\tau_2$ coincides with the completion of the job of $\tau_4$ also for both cases. Note that task $\tau_3$ is not considered because, being a delaying task, it cannot preempt $\tau_4$.

**Table 1** Characteristics of task-set $\mathcal{T}_1$.

| task | $T_i$ | $C_i$ | $AJ_i$ | $\pi_i$ | $\theta_i$ |
|---|---|---|---|---|---|
| $\tau_1$ | 8 | 2 | 4 | 4 | 4 |
| $\tau_2$ | 10 | 2 | 1 | 3 | 3 |
| $\tau_3$ | 20 | 1 | 3 | 2 | 2 |
| $\tau_4$ | 40 | 12 | 2 | 1 | 2 |

The hyperperiod is $P^{\mathcal{T}_1} = 40$ and $U^{\mathcal{T}_1} = 0.8$.

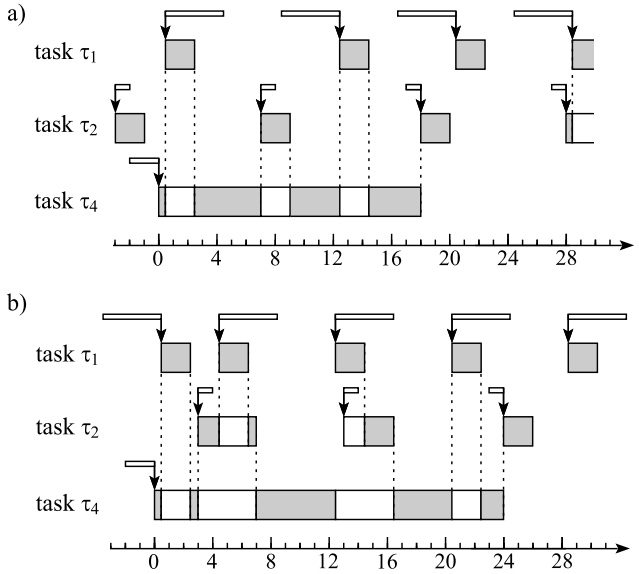


**Fig. 3** Two timelines for $\mathcal{T}_1 \setminus \{\tau_3\}$ showing the minimum (top) and maximum (bottom) hold times for a job of $\tau_4$ when experiencing extra preemptions by $\tau_1$ and minimal preemptions by $\tau_2$.

## 5.2 Intermediate hold times

We propose to evaluate all intermediate hold times in the range $[h^{\mathrm{mi}}, h^{\mathrm{ma}}]$ to determine the remaining values in the set $\mathcal{H}_i(\mathcal{E}_i)$. It is possible to evaluate all intermediate values because the number of preemptions $\eta_p$ that a preempting task $\tau_p$ causes in a job of a task $\tau_i$ is integer and finite. Furthermore, this number of preemptions that a job of $\tau_i$ experiences is bounded as follows:

$$\begin{aligned} \left\lceil \frac{h^{\mathrm{mi}} - AJ_e}{T_e} \right\rceil^* \le \eta_e \le \left\lceil \frac{h^{\mathrm{ma}} + AJ_e}{T_e} \right\rceil \quad & \text{for } \tau_e \in \mathcal{E}_i, \\ \left( \left\lceil \frac{h^{\mathrm{mi}} - AJ_m}{T_m} \right\rceil - 1 \right)^+ \le \eta_m \le \left( \left\lceil \frac{h^{\mathrm{ma}} - AJ_m}{T_m} \right\rceil - 1 \right)^+ \quad & \text{for } \tau_m \in \mathcal{M}_i \end{aligned} \tag{14}$$

The process of finding intermediate hold times in the set $\mathcal{H}_i(\mathcal{E}_i)$ therefore starts by choosing a candidate number of preemption $\eta_p$ in the range given in (14) for each preempting task. Given these number of preemptions, the corresponding values for $\beta_E$ and $\beta_M$ can alternatively be derived as follows:

$$\beta_E = \sum_{e:\tau_e \in \mathcal{E}_i} \eta_e \cdot BC_e, \qquad \beta_M = \sum_{m:\tau_m \in \mathcal{M}_i} \eta_m \cdot BC_m. \tag{15}$$

Afterwards, the candidate hold time $h^{\mathrm{ca}} \in [h^{\mathrm{mi}}, h^{\mathrm{ma}}]$ that we wish to evaluate is simply found using Equation (11). To find whether $h^{\mathrm{ca}}$ is a valid candidate, we first introduce the notion of *hold interval*.

**Definition 6** The *hold interval* $HI_i(y, \mathcal{E}_i)$ is defined as the hold time of a job $\iota_{i,k}$ of an artificial task $\tau_i'$ with a computation time $y \in \mathbb{R}^+$ when such a job $\iota_{i,k}$ experiences extra preemptions by the tasks in $\mathcal{E}_i$ according to the properties of optimal instant described in Theorem 1. In addition, all jobs in the interval assume their best-case computation times.

The hold interval $HI_i(y, \mathcal{E}_i)$ is given by the smallest $x \in \mathbb{R}^+$ satisfying the following recursive equation:

$$x = y + \sum_{e:\tau_e \in \mathcal{E}_i} \left\lceil \frac{x + g_e(x)}{T_e} \right\rceil^* BC_e, \tag{16}$$

where function $g_e(x)$ is defined as follow:

$$g_e(x) = \begin{cases} \alpha_e & \text{if } \left\lceil \dfrac{x+\alpha_e}{T_e} \right\rceil \leq \eta_e \\ \alpha_e - AJ_e & \text{otherwise.} \end{cases} \tag{17}$$

The activation delay $\alpha_e$ of an extra preempting task can be derived by Equation (**??**) using the chosen number of preemptions $\eta_e$ and the candidate hold time $h^{\text{ca}}$.

Recall that Equation (16) can be solved by an iterative procedure starting with a lower bound. Moreover, the result of $HI_i(y, \mathcal{E}_i)$ increases in every iteration till we arrive to a fixed point. During this iterative procedure, function $g(x)$ returns $\alpha_e$ when there are $\eta_e$ or less jobs of task $\tau_e$ preempting the hold time given by the intermediate result of $HI_i(y, \mathcal{E}_i)$. Therefore, these jobs of $\tau_e$ have minimal activation delay as described by Theorem 1. Furthermore, for latter jobs of $\tau_e$, function $g(x)$ returns $\alpha_e - AJ_e$ which means that these jobs have a maximal activation delay.

In order to find whether $h^{\text{ca}}$ is a valid candidate, the following equations must hold:

$$\begin{aligned} h^{\text{ca}} &= HI_i(\beta_M + BC_i, \mathcal{E}_i), \\ h^{\text{ca}} &= BH_i(\beta_E + BC_i, \mathcal{M}_i). \end{aligned} \tag{18}$$

If both equations in (18) hold, then $h^{\text{ca}}$ is a valid hold time for a job of $\tau_i$ and it is contained in the set $\mathcal{H}_i(\mathcal{E}_i)$. We can repeat this process evaluating candidates of hold times until we have explored all possible number of preemptions of preempting tasks bounded by (14).

As an example, consider again the set of tasks $\mathcal{T}_1$ described in Table 1. Let $\tau_1$ be an extra preempting task and $\tau_2$ a minimal preempting task of $\tau_4$. Recall that the minimum and maximum hold times for a job of $\tau_4$ are $h^{\text{mi}} = 18$ and $h^{\text{ma}} = 24$ respectively (see Figure 3). Based on this and using the bounds in (14), the number of preemptions $\eta_1$ and $\eta_2$ that tasks $\tau_1$ and $\tau_2$ cause in a job of $\tau_4$ is bounded by $2 \leq \eta_1 \leq 4$ and $1 \leq \eta_2 \leq 2$. Table 2 summarizes the result of the evaluation of the intermediate hold times considering the aforementioned bounds for $\eta_1$ and $\eta_2$. As can be seen, there are three hold times that satisfy the equations in (18). Those are the minimum and maximum hold times (trivial

case) and an intermediate hold time $h = 22$; therefore, $\mathcal{H}_4(\{\tau_1\}) = \{18, 22, 24\}$. Figure 4 shows a timeline in which such an intermediate hold time is assumed by a job of $\tau_4$.

**Table 2** Evaluation of intermediate candidates of hold times for task $\tau_4 \in \mathcal{T}_1$. A candidate hold time is considered valid when the set of equations in (18) hold.

| $\eta_1$ | $\eta_2$ | $\beta_1$ | $\beta_2$ | $h^{\text{ca}}$ | $\alpha_1$ | Valid |
|---|---|---|---|---|---|---|
| 2 | 1 | 4 | 2 | 18 | 2 | ✓ |
| 2 | 2 | 4 | 4 | 20 | 0 | ✗ |
| 3 | 1 | 6 | 2 | 20 | 4 | ✗ |
| 3 | 2 | 6 | 4 | 22 | 4 | ✓ |
| 4 | 1 | 8 | 2 | 22 | 4 | ✗ |
| 4 | 2 | 8 | 4 | 24 | 4 | ✓ |

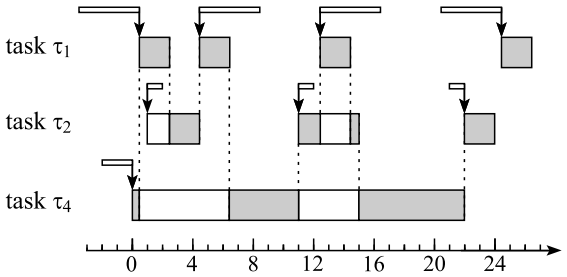


**Fig. 4** A timeline for $\mathcal{T}_1 \setminus \{\tau_3\}$ showing the intermediate hold time assumed by a job of $\tau_4$ when experiencing extra preemptions by $\tau_1$ and minimal preemptions by $\tau_2$.

## 6 Exact best-case response time analysis

In this chapter, we present and prove a novel exact best-case response time analysis for independent real-time periodic tasks with arbitrary deadlines scheduled using FPTS. Our analysis is based on the properties for an optimal instant presented in Theorem 1. More precisely, we first partition the set of preempting tasks into the sets of extra and minimal preempting tasks. For each set of extra preempting tasks of a task $\tau_i$, we determine the preemption vector $\boldsymbol{\eta}$ of a job $\iota_{i,k^{\text{opt}}}$ that experiences the properties for an optimal instant and subsequently derive its shortest response time. Afterwards, the best-case response time is simply determined by the minimum of all shortest response times for all partitions of preempting tasks.

### 6.1 Best-case interval for the execution of a task $\tau_i$

Similar to the best-case response time analysis with arbitrary deadlines for FPPS, a job of a task $\tau_i$ scheduled using FPTS and experiencing its optimal instant may still experience interference by its previous jobs provoking a delay on its start time (see Figure **??**). Therefore, we have to look to previous jobs of $\tau_i$ to determine its shortest response time. In order to do so, we have to determine intervals of minimal length with enough processing time to execute complete jobs of $\tau_i$.

**Definition 7** Let $\iota_{i,k}$ be a job of a task $\tau_i$ that experiences the properties for an optimal instant. Furthermore, let $\boldsymbol{\eta} \in \mathcal{H}_i(\mathcal{E}_i)$ be its corresponding preemption vector, where $\mathcal{E}_i$ is a set of extra preempting tasks of $\tau_i$. The *generalized best-case interval* $GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$ is defined as the length of the shortest interval $[t_s, t_e)$ before the completion of $\iota_{i,k}$, i.e. $t_e = f_{i,k}$, in which exactly an amount of time $y \in \mathbb{R}^+$ is available for the execution of task $\tau_i$.

Note that $GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$ is similar to the notion of best-case interval $BI_i(y)$ given in [**?**]. The main difference is that, for $GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$, it is considered that the interval ends with a job $\iota_{i,k}$ with a preemption vector $\boldsymbol{\eta}$. It is necessary to specify the preemption vector of $\iota_{i,k}$ in order to derive its hold time $H_{i,k} = h_i(\boldsymbol{\eta})$ and to determine the activation delays of its extra preempting tasks. We propose the following theorem for $GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$.

**Theorem 2** *The generalized best-case interval $GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$ is given by the largest $x \in \mathbb{R}^+$ satisfying*

$$x = y + \sum_{m:\tau_m \in \mathcal{M}_i} \left( \left\lceil \frac{x - \alpha_m(\boldsymbol{\eta})}{T_m} \right\rceil - 1 \right)^+ BC_m + \sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left\lfloor \frac{x - h_i(\boldsymbol{\eta}) - \alpha_\epsilon(\boldsymbol{\eta})}{T_\epsilon} \right\rfloor^+ BC_\epsilon + \beta_E(\boldsymbol{\eta}) \tag{19}$$

*where $\beta_E$ is the amount of time that extra preempting tasks preempts to $\iota_{i,k}$, i.e. $\beta_E(\boldsymbol{\eta}) = \sum_{e:\tau_e \in \mathcal{E}_i} \eta_e \cdot BC_e$. Furthermore, $\alpha_j(\boldsymbol{\eta})$ is a shorthand notation for $\alpha_j(\boldsymbol{\eta}, 0)$.*

*Proof* The general best-case interval $GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$ consists of two parts: the amount of time $y \in \mathbb{R}^+$ that corresponds to the first term in the RHS of Equation (19), and the interference that higher priority tasks induce in the interval $[t_s, t_e)$ of length $GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$. Since the set $\mathcal{M}_i$ contains the minimal preempting tasks of $\tau_i$, their influence on the interval $[t_s, t_e)$ must be minimal, and it is obtained when there is a simultaneous activation of all minimal preempting tasks at time $t_e$ with maximal activation delay. Therefore, the shortest amount of time reserved for minimal preempting tasks in the interval is given by

$$\sum_{m:\tau_m \in \mathcal{M}_i} \left( \left\lceil \frac{t_e - t_s - AJ_m}{T_m} \right\rceil - 1 \right)^+ BC_m = \sum_{m:\tau_m \in \mathcal{M}_i} \left( \left\lceil \frac{GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i) - \alpha_m(\boldsymbol{\eta})}{T_m} \right\rceil - 1 \right)^+ BC_m.$$

This corresponds to the second term in Equation (19).

Note that Definition 7 assumes that a job $\iota_{i,k}$ of task $\tau_i$ with a preemption vector $\boldsymbol{\eta}$ executes at the end of $[t_s, t_e)$, i.e. $f_{i,k} = t_e$ and $s_{i,k} = t_e - h_i(\boldsymbol{\eta})$. The amount of time reserved for extra preempting tasks in the interval $[s_{i,k}, t_e)$ of size $h_i(\boldsymbol{\eta})$ is given by the number of extra preempting jobs that execute in such an interval; hence, it is given by $\beta_E(\boldsymbol{\eta})$. This corresponds to the last term in the RHS of Equation (19). On the other hand, delaying tasks cannot preempt $\tau_i$; hence, there is no time reserved for delaying tasks in $[s_{i,k}, t_e)$. It only remains to determine the amount of interference by extra preempting and delaying tasks in the interval $[t_s, s_{i,k})$.

From the properties for an optimal instant given in Theorem 1, a simultaneous activation of delaying and extra preempting tasks occurs at time $t = s_{i,k} + \Delta$. Furthermore, the activation delay of delaying and extra preempting tasks at that time has to be maximal while preserving the preemption vector $\boldsymbol{\eta}$ for $\iota_{i,k}$, i.e. the activation delay of a task $\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i$ is given by $\alpha_\epsilon(\boldsymbol{\eta}, \Delta)$. Note that delaying tasks activated before $s_{i,k}$ have to be completed before $s_{i,k}$ as well. If not, job $\iota_{i,k}$ would not be able to start at that time. Hence, all jobs of delaying tasks activated in the interval $[t_s, s_{i,k})$ will give rise to interference in such an interval. The amount of processing time spent on the execution of delaying and extra preempting tasks in $[t_s, s_{i,k})$ is therefore given by

$$\sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left( \left\lceil \frac{(s_{i,k} + \Delta) - t_s - \alpha_\epsilon(\boldsymbol{\eta}, \Delta)}{T_\epsilon} \right\rceil - 1 \right)^+ BC_\epsilon =$$

$$\sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left( \left\lceil \frac{t_e - h_i(\boldsymbol{\eta}) + \Delta - t_s - \alpha_\epsilon(\boldsymbol{\eta}, \Delta)}{T_\epsilon} \right\rceil - 1 \right)^+ BC_\epsilon =$$

$$\sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left( \left\lceil \frac{GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i) - h_i(\boldsymbol{\eta}) + \Delta - \alpha_\epsilon(\boldsymbol{\eta}, \Delta)}{T_\epsilon} \right\rceil - 1 \right)^+ BC_\epsilon$$

Given Lemma **??** in the Appendix, when $\Delta$ approaches zero the equation is simplified as follows.

$$\lim_{\Delta \downarrow 0} \sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left( \left\lceil \frac{GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i) - h_i(\boldsymbol{\eta}) + \Delta - \alpha_\epsilon(\boldsymbol{\eta}, \Delta)}{T_\epsilon} \right\rceil - 1 \right)^+ BC_\epsilon =$$

$$\sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left( \lim_{\Delta \downarrow 0} \left\lceil \frac{GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i) - h_i(\boldsymbol{\eta}) + \Delta - \alpha_\epsilon(\boldsymbol{\eta}, \Delta)}{T_\epsilon} \right\rceil - 1 \right)^+ BC_\epsilon =$$

{Lemma **??**}

$$\sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left\lfloor \lim_{\Delta \downarrow 0} \frac{GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i) - h_i(\boldsymbol{\eta}) + \Delta - \alpha_\epsilon(\boldsymbol{\eta}, \Delta)}{T_\epsilon} \right\rfloor^+ BC_\epsilon =$$

$$\sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left\lfloor \frac{GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i) - h_i(\boldsymbol{\eta}) - \alpha_\epsilon(\boldsymbol{\eta})}{T_\epsilon} \right\rfloor^+ BC_\epsilon.$$

This corresponds to the third term in Equation (19).

$GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$ can be found by an iterative procedure starting with an upper bound, e.g. the worst-case response time of $\tau_i$ for a computation time $y$. It is worth noting that the generalized best-case interval $GI_i(y, \boldsymbol{\eta}, \mathcal{E}_i)$ specializes to

the best-case interval $BI_i(y)$ for FPPS when there are no delaying and extra preempting tasks.