Towards BCRT for FPTS
Exact analysis

H.J. Rivera Verduzco 0977393

P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Email: H.J.Rivera.Verduzco@student.tue.nl

May 23, 2017

1 Exact best-case analysis for FPTS

1.1 Optimal instant

Based on Fact 3, we can distinguish between two types of tasks than can influence the *best-case response time* of a task τ_i. These types of tasks are the set of *preemptive* tasks $\mathcal{H}_i = \{\tau_h | \pi_h > \theta_i\}$, and the set of *delaying* tasks $\mathcal{D}_i = \{\theta_i \geq \pi_d > \pi_i\}$. Furthermore, Fact 5 shows that for some cases the *best-case response time* of a task τ_i is not necessarily found in the job experiencing the *best-case hold time*. Therefore, the activation of all *preemptive* tasks may not coincide with the completion of the job k^{bcrt} of τ_i that assumes the *best-case response time*. Instead, some *preemptive* tasks may give rise to extra preemptions in k^{bcrt} in order to increase the *hold time* $H_{i,k^{bcrt}}$ and subsequently reduce the *response time* $R_{i,k^{bcrt}}$. Based on the previous observation, we divide the set of *preemptive* tasks $\mathcal{H}_i$ of a task τ_i into the set of *preemptive* tasks that give rise to extra preemptions denoted as $\mathcal{E}_i \subseteq \mathcal{H}_i$, and the remaining preemptive tasks that we call *best preemptive* tasks in the set $\mathcal{P}_i = \mathcal{H}_i \setminus \mathcal{E}_i$.

We now formulate the notion of optimal instant for FPTS based on the set of tasks influencing the *best-case response time* of a task τ_i.

Theorem 1. *In FPTS, an optimal instant for a task τ_i occurs when the completion of a job $\iota_{i,k}$ of τ_i coincides with a simultaneous activation of all best preemptive tasks $\tau_p \in \mathcal{P}_i$. Furthermore, the activation of all delaying tasks $\tau_d \in \mathcal{D}_i$ and all extra preemptive tasks $\tau_e \in \mathcal{E}_i$, coincides with $s_{i,k} + \Delta$, where Δ is a sufficiently small amount of time*

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 τ_i is equal to its priority. Furthermore, the set of *extra preemptive* tasks $\mathcal{E}_i$ is always empty when there are no *delaying* tasks. Therefore, we conclude that the optimal instant for a task τ_i scheduled under FPTS described in Theorem 1 specializes to an optimal instant for FPPS when $\theta_i = \pi_i$.

Figure 1 shows an example of an optimal instant under FPTS for task τ_i of the task-set $\mathcal{T}_1$ described in Table 1. As can be seen, the activation of the *best preemptive* task τ_{h1} coincides with the completion of task τ_i at time $t = 0$. Furthermore, *delaying* task τ_d and *extra* preemptive task τ_{h2} are activated an instant Δ after the start of τ_i.

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

	T_i	C_i	π_i	θ_i	BR_i
τ_{h1}	35	5	4	4	5
τ_{h2}	35	5	3	3	5
τ_d	50	20	2	2	20
τ_i	70	22	1	2	27

The *least common multiple* of the periods is 350 and $U^{\mathcal{T}_1} = 1$.

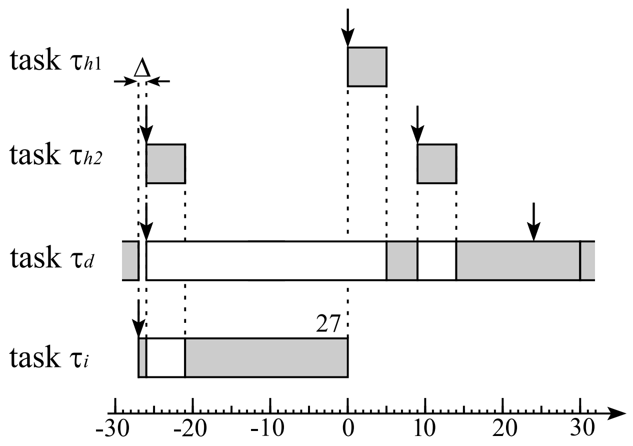

Figure 1: Optimal instant for task $\tau_i \in \mathcal{T}_1$ under FPTS when $\mathcal{E}_i = \{\tau_{h2}\}$ and $\mathcal{P}_i = \{\tau_{h1}\}$.

It is worth noting that, from the notion of optimal instant given above, it is not clear whether a *preemptive* task should belong to the set of *extra preemptive* tasks or to the set of *best preemptive* tasks. In fact, we could have chosen $\mathcal{E}_i = \{\tau_{h1}\}$ and $\mathcal{P}_i = \{\tau_{h2}\}$ in the example depicted in Figure 1, and we would still have obtained an option that is in accordance with the notion of optimal instant given in Theorem 1. Therefore, we can 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 τ_i. In the worst of the cases, these candidates of optimal instant are determined based on all possible combinations of *extra preemptive* and *best preemptive* tasks. Clearly, this differs from the best-case analysis for FPPS, where the optimal instant is always unique.

1.2 Determining the phasing of delaying and extra preemptive tasks

Assuming that we know the set of *extra preemptive* (and *best preemptive*) tasks that lead to the optimal instant where the *best-case response time* for a task τ_i is found, we now are interested in determining the phasing for each task in order to calculate such a *best-case response time*. According to Theorem 1, the simultaneous activation of *best-preemptive* tasks has to take place at time $t_1 = f_{i,k^{bcrt}}$, where k^{bcrt} is the job of τ_i that assumes the best-case response time. Furthermore, the simultaneous activation of *delaying* and *extra preemptive* tasks occurs at time $t_2 = s_{i,k^{bcrt}} + \Delta$. Therefore, the phasing ϕ_r for *delaying* and *extra preemptive* tasks relative to the activation of *best-preemptive* tasks is given by

$$\phi_r = t_2 - t_1 = (s_{i,k^{bcrt}} + \Delta) - f_{i,k^{bcrt}} = -H_{i,k^{bcrt}} + \Delta.$$

We therefore have to determine hold-time $H_{i,k^{bcrt}}$ to derive the proper phasing for *delaying* and *extra preemptive* tasks where we can find the *best-case response time* of task τ_i. Furthermore, note that $H_{i,k^{bcrt}}$ is only influenced by *preemptive* tasks in $\mathcal{E}_i$ and $\mathcal{P}_i$ because *delaying tasks* cannot preempt τ_i. In the rest of this section, we introduce the notions of *worst-case* and *best-case* hold intervals. In addition, we determine the hold time $H_{i,k^{bcrt}}$ based on such notions.

Definition 1. *The worst-case hold interval $WI_i(y,\mathcal{E})$ is defined as the worst-case hold time of an artificial task τ'_i with computation time of $y \in \mathbb{R}^+$ in task-set $\mathcal{E} \cup \{\tau'_i\}$ given that all jobs in the interval assume their best-case computation times.*

The *worst-case hold interval* follows directly from the formula for *worst-case hold time* for FPTS given in [2]. The only difference is that best-case computation times are assumed for the *worst-case hold interval*. Hence, $WI_i(y,\mathcal{E})$ is given by the smallest $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{e:\pi_e > \theta_i, \tau_e \in \mathcal{E}} \left\lceil \frac{x}{T_e} \right\rceil BC_e. \tag{1}$$

Definition 2. *The best-case hold interval $BI_i(y,\mathcal{P})$ is defined as the best-case hold time of an artificial task τ'_i with computation time of $y \in \mathbb{R}^+$ in task-set $\mathcal{P} \cup \{\tau'_i\}$ given that $\mathcal{P}$ contains only preemptive tasks.*

Since Definition 2 assumes that $\mathcal{P}$ only contains *preemptive* tasks, we can use the same formula for *best-case hold time* for FPPS given in [3] to calculate $BI_i(y,\mathcal{P})$. Hence, function $BI_i(y,\mathcal{P})$ returns the largest positive solution of $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{p:\pi_p > \theta_i, \tau_p \in \mathcal{P}} \left(\left\lceil \frac{x}{T_p} \right\rceil - 1 \right)^+ BC_p. \tag{2}$$

Finally, we introduce a function to determine the hold-time of a task τ_i based on its *extra preemptive* and *best-preemptive* tasks.

Definition 3. *The hold time $HT_i(\mathcal{E})$ is defined as the shortest hold time of a job $\iota_{i,k}$ of $\tau_i \in \mathcal{T}$ when experiencing extra preemptions by the tasks in $\mathcal{E} \subset \mathcal{T}$. This is, when the tasks in $\mathcal{E}$ are simultaneously activated at time $s_{i,k} + \Delta$. Where Δ is a sufficiently small amount of time. Furthermore, delaying tasks of τ_i in $\mathcal{T}$ are ignored.*

$HT_i(\mathcal{E})$ is given by the following equation.

$$HT_i(\mathcal{E}) = \beta_p + \beta_e + BC_i, \tag{3}$$

where $\beta_p, \beta_e \in \mathbb{R}^+ \cup \{0\}$ are the influence that *best preemptive* and *extra preemptive* tasks of τ_i have in its hold time respectively. The values of β_p and β_e are given by the smallest solution of the following recursive equations:

$$\begin{aligned} \beta_e &= WI_i(\beta_p + BC_i, \mathcal{E}) - \beta_p - BC_i, \\ \beta_p &= BI_i(\beta_e + BC_i, \mathcal{P}) - \beta_e - BC_i, \end{aligned} \tag{4}$$

where $\mathcal{P} = \{\tau_p | \pi_p > \theta_i, t_p \notin \mathcal{E}\}$ is the set of *best preemptive* tasks. β_p and β_e can be found by an iterative procedure starting with a lower bound.

Figure 2 shows a timeline depicting $HT_i(\mathcal{E}) = 14$ for $\tau_i \in \mathcal{T}_2$ with characteristics as described in Table 2. As can be seen, task τ_e is selected as the *extra preemptive* task, i.e. $\mathcal{E} = \{\tau_e\}$, while τ_p is a *best preemptive* task. Note that an activation of τ_p coincides with the completion of the job of τ_i; therefore, minimizing its influence on the hold time of τ_i. In addition, since tasks τ_e and τ_p preempt task τ_i three times each, $\beta_e = 6$ and $\beta_p = 3$. Finally, note that *delaying* task τ_d is not considered in Figure 2 because Definition 3 assumes that *delaying* tasks are ignored.

Table 2: Characteristics of task-set $\mathcal{T}_2$.

	T_i	C_i	π_i	θ_i
τ_p	4	1	4	4
τ_e	5	2	3	3
τ_d	10	1	2	2
τ_i	20	5	1	2

The *least common multiple* of the periods is 20 and $U^{\mathcal{T}_2} = 1$.

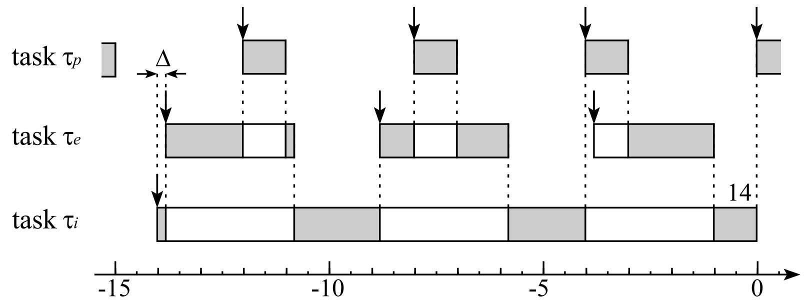

Figure 2: Timeline for $\mathcal{T}_2 \setminus \{\tau_d\}$ depicting a job of task τ_i that assumes a hold time of $HT_i(\mathcal{E}_i) = 14$ where $\mathcal{E}_i = \{\tau_e\}$. For this example, $\beta_e = 6$ and $\beta_p = 3$.

We now formulate a corollary for the phasing ϕ_r of *delaying* and *extra preemptive* tasks, based on the hold time of the job of task τ_i experiencing the optimal instant.

Corollary 1. *Let $\mathcal{E}_i^{br}$ be the set of extra preemptive tasks of a task τ_i that leads to its best-case response time. The phasing of delaying and extra preemptive tasks relative to the activation of best preemptive tasks that leads to an optimal instant is then given by $\phi_r = -HT(\mathcal{E}_i^{br}) + \Delta$.*

1.3 Best-case response time for FPTS

Similar to the *best-case response time* analysis with arbitrary deadlines for FPPS, a job of a task τ_i scheduled under FPTS and experiencing its optimal instant may still experience interference by its previous jobs. Therefore, we have to look to previous jobs of task τ_i to determine its *best-case response time*. In order to do so, we have to determine intervals of minimal length with enough processing time to execute complete jobs of τ_i.

Definition 4. *Given a job $\iota_{i,k}$ of a task τ_i with a hold time $H_{i,k} = HT_i(\mathcal{E}_i)$, where $\mathcal{E}_i$ is a set of extra preemptive tasks, the generalized best-case interval $GI_i(y, \mathcal{E}_i)$ is defined as the length of the shortest interval before the completion of $\iota_{i,k}$ in which an amount of time $y \in \mathbb{R}^+$ is available for the execution of τ_i.*

Note that $GI_i(y, \mathcal{E}_i)$ is similar to the notion of *best-case interval* given in [1]. The main difference is that, for $GI_i(y, \mathcal{E}_i)$, the interval is considered before the completion of a job of τ_i with hold time $HT_i(\mathcal{E}_i)$. It is necessary to specify the hold time in order to know the phasing at which *extra preemptive* and *delaying* tasks must be activated. Therefore, we propose the following corollary.

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

$$x = y + \sum_{p:\tau_p \in \mathcal{P}_i} \left(\left\lceil \frac{x}{T_p} \right\rceil - 1 \right)^+ BC_p + \sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left\lfloor \frac{x - HT_i(\mathcal{E}_i)}{T_\epsilon} \right\rfloor BC_\epsilon + \beta_e \tag{5}$$

where $\mathcal{P}_i = \{\tau_p | \pi_p > \theta_i, \tau_p \notin \mathcal{E}_i\}$ is the set of best preemptive tasks of τ_i, and $\mathcal{D}_i = \{\tau_d | \theta_i \geq \pi_d > \pi_i\}$ is the set of delaying tasks.

Proof. Since the set $\mathcal{P}_i$ contains the *best preemptive* tasks of τ_i, their influence on the interval of length $GI_i(y, \mathcal{E}_i)$ must be minimal, and it is obtained when the simultaneous activation of all *best preemptive* tasks coincides with the completion of job $\iota_{i,k}$. Therefore, the amount of time reserved for *best preemptive* tasks in the interval is simply given by $\sum_{p:\tau_p \in \mathcal{P}_i} \left(\left\lceil \frac{GI_i(y, \mathcal{E}_i)}{T_p} \right\rceil - 1 \right)^+ BC_p$. This corresponds to the second term in Equation 5.

Given Corollary 1, we know that the phasing of *delaying* and *extra preemptive* tasks relative to *best-preemptive* tasks is $\phi_r = -HT(\mathcal{E}_i) + \Delta$. We can simulate this phasing by injecting an artificial activation jitter of $-\phi_r$ to *delaying* and *extra preemptive* tasks as shown in the example depicted

Table 3: Task set $\mathcal{T}_3$.

task	T_i	C_i	π_i	θ_i
τ_p	35	5	4	4
τ_e	35	5	3	3
τ_d	50	20	2	2
τ_i	70	22	1	2

The *least common multiple* of the periods is 350 and $U^{\mathcal{T}_3} = 1$.

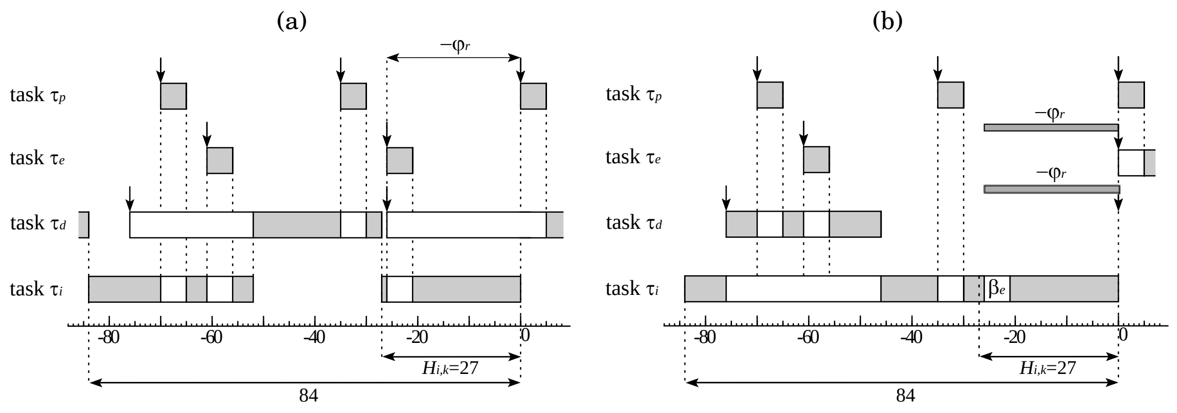

Figure 3: Two timelines for $\mathcal{T}_3$ depicting the *generalized best-case interval* $GI(2\cdot BC_i, \mathcal{E}_i) = 84$, where $\mathcal{E}_i = \{\tau_e\}$. (a) shows the interval under FPTS scheduling while (b) shows the same interval in the analogous representation using FPPS when injecting an artificial activation jitter of $-\phi_r$ to tasks τ_d and τ_e.

in Figure 3. Note that, by injecting this artificial jitter, we are not considering the interference of *delaying* and *extra preemptive* tasks in hold time $H_{i,k}$. Therefore, the amount of processing time spent on the execution of *delaying* and *extra preemptive* tasks before the start of job $\iota_{i,k}$ is given by

$$\sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left(\left\lceil \frac{GI_i(y, \mathcal{E}_i) - (-\phi_r)}{T_\epsilon} \right\rceil - 1 \right)^+ BC_\epsilon = \lim_{\Delta \to 0} \sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left(\left\lceil \frac{GI_i(y, \mathcal{E}_i) - HT_i(\mathcal{E}_i) + \Delta}{T_\epsilon} \right\rceil - 1 \right)^+ BC_\epsilon$$
$$= \sum_{\epsilon:\tau_\epsilon \in \mathcal{E}_i \cup \mathcal{D}_i} \left\lfloor \frac{GI_i(y, \mathcal{E}_i) - HT_i(\mathcal{E}_i)}{T_\epsilon} \right\rfloor BC_\epsilon.$$

This corresponds to the third term in Equation 5.

Finally, we have to consider the interference induced by *delaying* and *extra preemptive* tasks in the hold time of $\iota_{i,k}$. Clearly, this is simply given by β_e because *delaying* tasks cannot preempt task τ_i. □

Given the notion of *generalized best-case interval*, the *best-case response time* analysis for FPTS follows directly from the analysis for FPPS.

Theorem 2. *Let $\mathcal{E}_i^{br}$ be the set of extra preemptive tasks of a task $\tau_i \in \mathcal{T}$ that leads to its best-case response time. Furthermore, let wl'_i exists, where wl'_i is the worst-case number of jobs of τ_i in a level-i active period in task-set $\mathcal{T}_i = \{\tau_a | \pi_a \geq \pi_i, \tau_a \in \mathcal{T}\}$. The best-case response time of task τ_i scheduled under FPTS is given by*

$$BR_i = \max_{1 \leq k \leq wl'_i} (GI_i(k \cdot BC_i, \mathcal{E}_i^{br}) - (k-1)T_i). \tag{6}$$

It is worth noting that the number of jobs to explore in order to find the *best-case response time* is given by wl'_i in Theorem 2. Furthermore, wl'_i is determined using an auxiliary task-set $\mathcal{T}_i$ that contains all tasks of $\mathcal{T}$ with the exception of lower priority tasks of τ_i. The reason for this is that, as was proved elsewhere, lower priority tasks do not influence the *best-case response time* of a task τ_i. Therefore, the *best-case response time* of $\tau_i \in \mathcal{T}$ is the same as the *best-case response time* of the same task in $\mathcal{T}_i$. Based on this, we conclude that it is sufficient to explore the worst-case number of jobs of τ_i in a level-i active period in task-set $\mathcal{T}_i$ to determine the *best-case response time*.

From Theorem 2, we also observe that we need the set of *preemptive* tasks $\mathcal{E}_i^{br}$ in order to find the *best-case response time*. Since determining $\mathcal{E}_i^{br}$ is non-trivial as we will show in the next section, we propose the following theorem where all possible values for $\mathcal{E}_i^{br}$ are explored.

Theorem 3. *Let all tasks of a set $\mathcal{T}$ be strictly periodic and let wl'_i exists, where wl'_i is the worst-case number of jobs of τ_i in a level-i active period in task-set $\mathcal{T}_i = \{\tau_a | \pi_a \geq \pi_i, \tau_a \in \mathcal{T}\}$. Furthermore, let $\mathcal{H}_i' = \{\tau_h | \pi_h > \theta_i, \exists \tau_d : \theta_i \geq \pi_d > \pi_i\}$ be the set of preemptive tasks of a task $\tau_i \in \mathcal{T}$ when at least one delaying task exists. The best-case response time of task τ_i is given by*

$$BR_i = \min_{\mathcal{E} \in \hat{\mathcal{E}}_i}\Big(\max_{1 \leq k \leq wl'_i} (GI_i(k \cdot BC_i, \mathcal{E}) - (k-1)T_i)\Big), \tag{7}$$

where $\hat{\mathcal{E}}_i$ is the set of all possible combinations of $\mathcal{H}_i'$ including the empty set.

1.4 Exact BCRT analysis for FPTS is NP-Hard

In this section, we show that for some task-sets finding the exact best-case response time of a task scheduled using FPTS is NP-hard.

In FPTS, *delaying* tasks may provoke that the best-case response time of a task τ_i is assumed when some *preemptive* tasks are activated a sufficiently small amount of time after the start of the job k^{bcrt} of τ_i that assumes the best-case response time. We denoted this category of tasks as *extra preemptive* because they give rise to extra preemptions in job k^{bcrt}, whereas the *preemptive* tasks that interfere at least as possible with k^{bcrt} are called *best preemptive* tasks. Due to this distinction between *extra preemptive* and *best preemptive* tasks, the main concern to determine the best-case response time of a task is to identify which *preemptive* tasks correspond to which category. In other words, given the set $\mathcal{H}_i = \{\tau_h | \pi_h > \theta_i\}$ of *preemptive tasks*, we have to find the set $\mathcal{E}_i \subseteq \mathcal{H}_i$ of *extra preemptive* tasks that minimizes the response time of τ_i. The set of *best preemptive* tasks is simply given by $\mathcal{P}_i = \mathcal{H}_i \backslash \mathcal{E}_i$. In principle, this is an optimization problem; however, in many cases the set $\mathcal{E}_i$ can only be empty. For instance when the task τ_i is non-preemptive or when there are no *delaying* tasks, i.e. when $\pi_i = \theta_i$. On the other hand, there are other cases where determining the set of *extra preemptive* tasks is not trivial. Table 4 shows an example of such a nontrivial task-set. As we will show, determining the best-case response time of τ_i for this and similar cases is NP-hard.

Table 4: Task set $\mathcal{T}_4$.

task	T_i	$WC_i = BC_i$	π_i	θ_i
$\{\tau_{h1}, \tau_{h_2}, \tau_{h_3}, \tau_{h_4}, \tau_{h_5}\}$	$\{35,...,35\}$	$\{3.3, 2.3, 2, 1.3, 1.1\}$	$\{7,...,3\}$	$\{7,...,3\}$
τ_d	50	20	2	2
τ_i	70	22	1	2

The *least common multiple* of the periods is 350 and $U^{\mathcal{T}_4} = 1$.

Figure 4 shows a timeline for $\mathcal{T}_4$ where all *preemptive* tasks $\mathcal{H}_i$ are activated simultaneously with the completion of the last job $\iota_{i,1}$ of τ_i at time $t = 0$. Hence, in this timeline the set $\mathcal{E}_i$ is empty and the response time of the last job $\iota_{i,1}$ is $R_{i,1} = 36$. Furthermore, note that $R_{i,1}$ cannot be reduced by pushing the activations of τ_i because the job $\iota_{i,3}$ activated at time $t = -176$ starts upon activation. Hence, pushing the activation of τ_i would modify the schedule. In order to reduce the response time of $\iota_{i,1}$, the activations of τ_d should be pushed to an earlier moment in time till one of its activations occurs at time $t = -176$. In this way, τ_d will prevent job $\iota_{i,3}$ to start upon activation and, consequently, it will be possible to reduce $R_{i,1}$. However, forcing τ_d to be activated at time

$t = -176$ would lead to an activation of τ_d at time $t = -26$ as well. In order to avoid this activation to influence on the response time of $\iota_{i,1}$, its hold time has to be increased till $H_{i,1} > 26$.

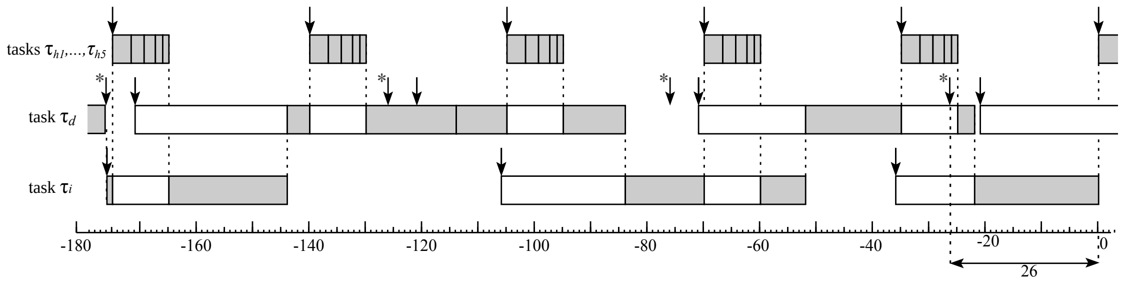

Figure 4: A timeline for $\mathcal{T}_4$ when the set of *extra preemptive* tasks $\mathcal{E}_i$ is empty. Down-arrows with a "star" denote the activations fo τ_d that would allow to reduce the response time of the last job $\iota_{i,1}$ of τ_i.

Since the only tasks that can increase the hold time of $\iota_{i,1}$ are the *preemptive* tasks, the problem reduces to find the *preemptive* tasks that minimizes the hold time of $\iota_{i,1}$ satisfying the constraint $H_{i,1} > 26$. This problem is an instance of the dual of the well known knapsack problem that has been shown to be NP-hard. For task-set $\mathcal{T}_4$, the solution of this problem is depicted in Figure 5, where the *extra preemptive* tasks that minimize the response time of $\iota_{i,1}$ are τ_{h_2} and τ_{h_3}. Therefore, the best-case response time of task τ_i is $BR_i = BC_i + BC_{h2} + BC_{h3} = 26.3$.

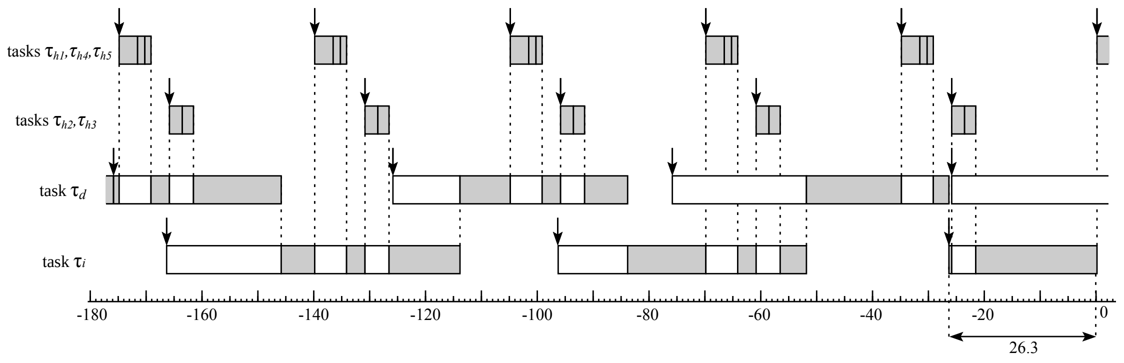

Figure 5: A timeline for $\mathcal{T}_4$ showing the best-case response time of τ_i with $BR_i = 26.3$. The set of *extra preemptive* tasks that leads to the best-case response time is $\mathcal{E}_i = \{\tau_{h_2}, \tau_{h_3}\}$.

1.5 An algorithm for computing BCRT for FPTS

The outer min in the *best-case response time* analysis for FPTS described in Theorem 3 requires to find the minimum response time among all possible combinations of *extra preemptive* tasks. This clearly suggests an algorithm with worst-case time complexity of $\mathcal{O}(2^n)$ for n number of tasks. Due to the inefficient nature of the analysis, in this section we propose an implementation for Theorem 3 that achieves a more tractable complexity on average; however, in the worst-case it is still exponential.

Algorithm 2 shows the exhaustive algorithm to compute the *best-case response time* of a task τ_i given a task-set $\mathcal{T}$. The algorithm first calls function $bcrtInit(\mathcal{T}, i)$ to initialize the set of possible *extra preemptive* tasks $\mathcal{H}_i'$ and a first candidate for the *best-case response time* BR_i. If $\mathcal{H}_i'$ results to be empty, then the algorithm terminates. Otherwise, the algorithm continues and try to find the minimum response time trying all possible combinations for the set of possible *extra preemptive* tasks.

Algorithm 3 shows the $bcrtInit(\mathcal{T},i)$ procedure. Line 2 first assigns to α the best-case hold time when ignoring all *delaying* tasks. Line 3 then checks whether the same hold time can be assumed when introducing *delaying* tasks. If not, the algorithm jumps to Lines 13 and 14 where the set of possible extra preemptive tasks $\mathcal{H}_i'$ is simply initialized with all the *preemptive* tasks. On the other hand, if the hold time can be assumed after introducing *delaying* tasks, there is a high probability that the job experiencing this hold time is the job with the *best-case response time*[1]. Therefore, the algorithm continues to Line 4 where it computes the shortest response time when there are no *extra preemptive* tasks. In Lines 5-7, it is determined the amount of time that the relative activation α of *delaying* tasks has to be increased in order to reduce the response time. If α results to be greater than the current response time, then there is no way that the response time can be decreased and we indeed have found the *best-case response time*[2]; hence, the set of possible extra preemptive tasks $\mathcal{H}_i'$ becomes empty in Line 9. Otherwise, the *preemptive* tasks that can increase the hold time without exceeding the current response time are added to the set of possible *extra preemptive* tasks in Line 11.

Algorithm 1 Exhaustive algorithm to derive *best-case response time* under FPTS.

```
1: procedure bcrtFPTS($\mathcal{T}$,$i$)
2:     $(BR_i, \mathcal{H}_i') \leftarrow bcrtInit(\mathcal{T}, i)$;
3:     if $\mathcal{H}_i' \neq \{\}$ then
4:         $\hat{\mathcal{E}} \leftarrow$ set of all possible combinations of $\mathcal{H}_i'$;
5:         for each $\mathcal{E}$ of $\hat{\mathcal{E}}$ do
6:             $BR_i \leftarrow \min\{BR_i, \max_{1 \le k \le wl_i'} (GI_i(k \cdot BC_i, \mathcal{E}) - (k-1)T_i)\}$;
           end for
       end if
7:     return $BR_i$;
```

Algorithm 2 Algorithm to derive an initial possible *best-case response time* and a set of candidates of *extra preemptive* tasks.

```
1: procedure bcrtInit($\mathcal{T}$,$i$)
2:     $\alpha \leftarrow HT_i(\{\})$;
3:     if $\alpha = GI_i(BC_i, \{\})$ then
4:         $BR_i^{init} \leftarrow \max_{1 \le k \le wl_i'} (GI_i(k \cdot BC_i, \{\}) - (k-1)T_i)$;
5:         $k^{tight} \leftarrow$ the smallest $k$ with $1 \le k \le wl_i'$ that leads to $BR_i^{init}$;
6:         $DI_i \leftarrow BR_i^{init} + (k^{tight} - 1)T_i - \alpha$;
7:         $\alpha \leftarrow \min\{\infty, \min_{d:\theta_i \ge \pi_d > \pi_i} (DI_i \mod T_d)\} + \alpha$;
8:         if $\alpha \ge BR_i^{init}$ then
9:             $\mathcal{H}_i' = \{\}$;
10:        else
11:            $\mathcal{H}_i' = \{\tau_h | \pi_h > \theta_i, BC_h < BR_i - \alpha, \tau_h \in \mathcal{T}\}$
           end if
12:    else
13:        $BR_i^{init} \leftarrow \infty$;
14:        $\mathcal{H}_i' = \{\tau_h | \pi_h > \theta_i, \tau_h \in \mathcal{T}\}$;
       end if
15:    return $BR_i^{init}, \mathcal{H}_i'$;
```

[1]Note: Right now this is just a claim, based on my observations and intuition. Not sure how to support this formally, yet.
[2]Note: Probably this has to be proved as well.

References

[1] R.J. Bril, J.J. Lukkien, and R.H. Mak. Best-case response times and jitter analysis of real-time tasks with arbitrary deadlines. In Proc. 21st International Conference on Real-Time Networks and Systems (RTNS), ACM, pp. 193-202, October 2013.

[2] R.J. Bril, S. Altmeyer, M.M.H.P. van den Heuvel, R.I. Davis, and M. Behnam. Fixed priority scheduling with pre-emption thresholds and cache-related pre-emption delays: integrated analysis and evaluation. In Real-Time Systems, 31 January 2017. doi:10.1007/s11241-016-9266-z

[3] R.J. Bril, G. Fohler, and W.F.J. Verhaegh. Execution times and execution jitter of real-time tasks under fixed-priority preemptive scheduling. Technical Report CSR 08-27, TU/e, The Netherlands, Oct. 2008. http://www.win.tue.nl/~mholende/cantata/publications/BCG-WiP-ECRTS09-final.pdf