Towards BCRT for FPTS

H.J. Rivera Verduzco 0977393

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

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

April 25, 2017

1 Recap on initial results

Fact 1. *In general, the optimal instant as presented for FPPS is not a valid optimal instant for FPTS.*

This fact illustrates the need of finding a generalized version of the optimal instant to find the *best-case response time* under FPTS.

Fact 2. *Under FPTS, the shortest response time of a task τ_i in a level-i active period is not necessarily assumed by the last job in that level-i active period.*

This fact illustrates that it is also necessarily to explore previous jobs in a level-i active period, not only the one experiencing the optimal instant.

Fact 3. *The best-case response time of a task τ_i can be affected by a higher priority task that cannot preempt τ_i.*

This facts shows the importance of considering delaying tasks in the *best-case response time analysis*

Table 1: Task set $\mathcal{T}_1$. The *least common multiple* of the periods is 350.

	T_i	$WC_i = BC_i$	π_i	θ_i	wl_i
τ_1	35	10	3	3	1
τ_2	50	20	2	2	2
τ_3	70	22	1	2	5

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

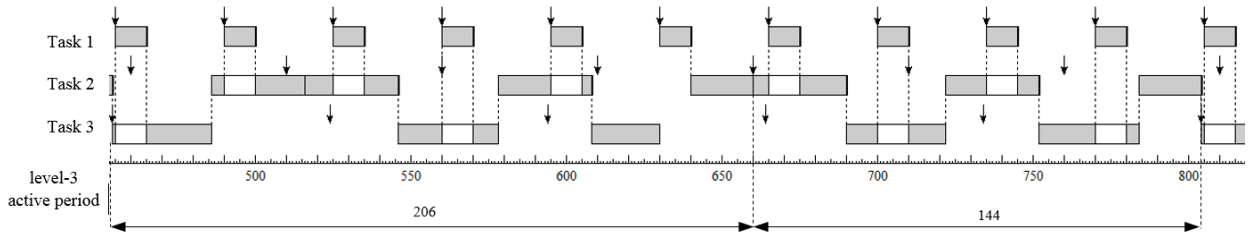

Figure 1: The shortest response time of task τ_3 occurs in the first job of the first level-3 active period.

Fact 4. *A task τ_l with a lower priority than a task τ_i can also affect the best-case response time of τ_i whenever τ_l's threshold θ_l is larger than or equal to τ_i's priority π_i, i.e. $\theta_i \geq \pi_i$.*

Further experiments suggest that this fact is probably not true. Probably a lower priority blocking task cannot affect the *best-case response time* of a higher priority task τ_i.

2 Further results

Fact 5. *In general, the best-case response time is not necessarily found when the activation of all higher priority tasks that can preempt a task τ_i coincides with the completion of a job of τ_i.*

Consider the taskset presented in Table 2. In Figure 2 a schedule is shown where the activation of higher priority tasks τ_1 and τ_2 coincides with the completion of a job of τ_4 at time $t = 736$. As can be seen, the shortest response time for τ_4 under this schedule is assumed by the job activated at time $t = 560$ with a response time of 32. However, this is not the *best-case response time* for τ_4. Figure 3 shows a schedule for $\mathcal{T}_2$ where a response time of $R_4 = 27$ for task τ_4 is found at time $t = 709$.

Table 2: Task set $\mathcal{T}_2$.

	T_i	$WC_i = BC_i$	π_i	θ_i	wl_i	BR_i
τ_1	35	5	4	4	1	5
τ_2	35	5	3	3	1	5
τ_3	50	20	2	2	2	20
τ_4	70	22	1	2	5	27

The *least common multiple* of the periods is 350 and $U^{\mathcal{T}_2} = 1$.

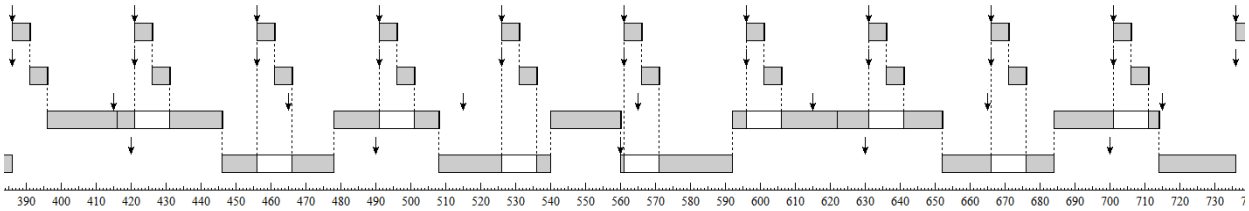

Figure 2: Simultaneous release of preemptive tasks coincides with the completion of a job of task τ_4 at time 736. The phases used to construct this schedule are $\phi_1 = \phi_2 = 1$, $\phi_3 = 15$, and $\phi_4 = 0$.

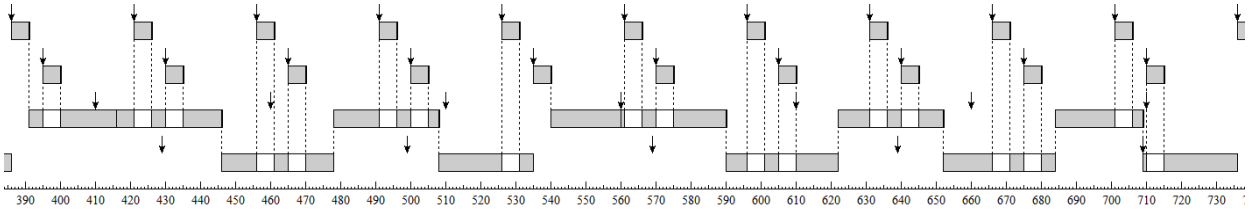

Figure 3: Schedule for $\mathcal{T}_2$ where the *best-case response time* ($BR_4 = 27$) for τ_4 is assumed at time $t = 709$. The phases used to construct this schedule are $\phi_1 = 1$, $\phi_2 = \phi_3 = 10$, and $\phi_4 = 9$.

Fact 6. *A higher priority task that cannot preempt a task τ_i can provoke that the completion of task τ_i could never occur simultaneously with the release of all its higher priority preemptive tasks.*

Consider the task-set $\mathcal{T}_3$ presented in Table 3. Figure 4 shows a schedule only considering tasks τ_1 and τ_3 of $\mathcal{T}_3$ where the activation of the highest priority task τ_1 coincides with the completion of task τ_3. However, this situation cannot be assumed when considering also τ_2. Figure 5 shows that as soon as τ_2 is introduced in the schedule, task τ_3 will always experience one preemption by τ_1. Note that the first job of τ_2 in Figure 5 delays the start time of the first job of τ_3; hence, causing the completion of τ_3 to not occur simultaneously with the activation of τ_1.

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

	T_i	$WC_i = BC_i$	π_i	θ_i	wl_i	WR_i^{T}	BR_i^{T}	RJ_i^{T}	WR_i^{P}	BR_i^{P}	RJ_i^{P}
τ_1	80	20	3	3	1	20	20	0	20	20	0
τ_2	30	15	2	2	8	104	15	89	35	15	20
τ_3	240	50	1	2	1	120	70	50	230	165	65

The *least common multiple* of the periods is 240 and $U^{\mathcal{T}_3} \approx 0.96$.

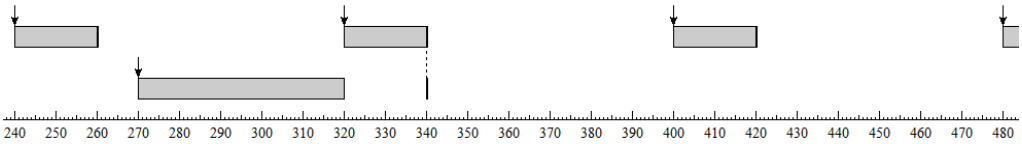

Figure 4: Schedule of task-set $\mathcal{T}_3 \setminus \{\tau_2\}$. The phases used to construct this schedule are $\phi_1 = 0$ and $\phi_3 = 30$.

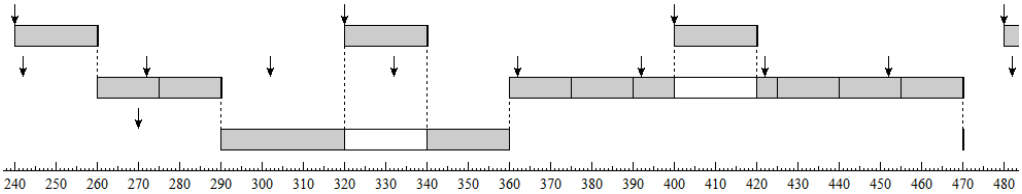

Figure 5: Schedule of task-set $\mathcal{T}_3$. The phases used to construct this schedule are $\phi_1 = 0$, $\phi_2 = 2$ and $\phi_3 = 30$.

2.1 Blocking tasks

Fact 7. *A lower priority task that can block the execution of a task τ_i cannot affect the best-case response time of τ_i.*

Consider task-set $\mathcal{T}_4$ described in Table 4. Figure 6 shows an schedule for such a task-set, but only considering tasks τ_1 and τ_2. As can be seen, the *best-case response time* $BR_2 = 50$ for task τ_2 is assumed in the schedule. Introducing the blocking task τ_3 could affect the response time of task τ_2 as shown in Figure 7. However, it is always possible to re-schedule a blocking task in such a way that it does not affect the *best-case response time* of its higher priority tasks. Figure 8 shows the task-set $\mathcal{T}_4$ scheduled in such a way that the *best-case response time* for task τ_2 is assumed.

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

	T_i	$WC_i = BC_i$	π_i	θ_i	wl_i	BR_i
τ_1	80	20	3	3	1	20
τ_2	240	50	2	2	1	50
τ_3	30	15	1	2	8	15

The *least common multiple* of the periods is 240 and $U^{\mathcal{T}_2} \approx 0.96$.

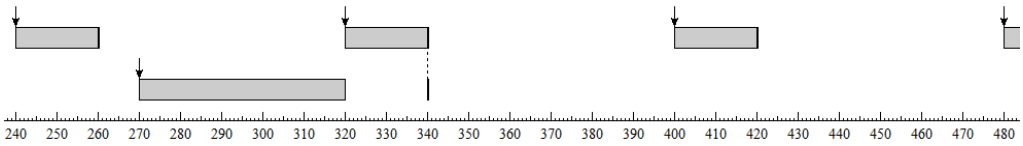

Figure 6: Schedule of taskset $\mathcal{T}_4 \backslash \tau_3$. The phases used to construct this schedule are $\phi_1 = 0$ and $\phi_2 = 30$.

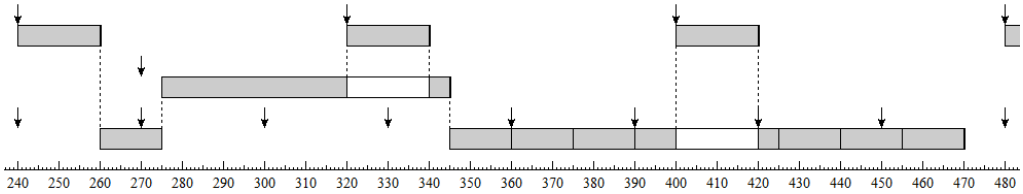

Figure 7: Schedule of task-set $\mathcal{T}_4$. The phases used to construct this schedule are $\phi_1 = \phi_3 = 0$ and $\phi_2 = 30$.

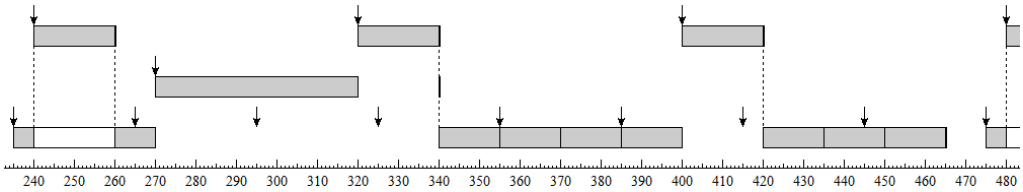

Figure 8: Schedule of task-set $\mathcal{T}_4$ where the *best-case response time* for task τ_2 is assumed. The phases used to construct this schedule are $\phi_1 = 0$, $\phi_3 = 25$ and $\phi_2 = 30$.

3 BCRT lower bound for FPTS

3.1 Hypothetical best-case interval for execution of a task τ_i

Definition 1. *The hypothetical best-case interval $HI_i(y,\alpha)$ is defined as the length of the shortest interval $[t_s,t_e)$ in which an amount of time $y \in \mathbb{R}^+$ is available for the execution of τ_i assuming that all jobs of delaying tasks activated at or after $t_e - \alpha$ cannot start, where $\alpha \in \mathbb{R}^+ \cup \{0\}$. Furthermore, delaying tasks activated before $t_e - \alpha$ can preempt τ_i.*

Note that $HI_i(y,\alpha)$ assumes that all higher priority tasks can preempt τ_i. Therefore, $HI_i(y,\alpha)$ can be seen as an extension of the *best-case interval* $BI_i(y)$ defined in [1], with the addition that jobs of *delaying* tasks activated after or at $t_e - \alpha$ are simply ignored. Hence, such jobs do not influence the amount of time available for task τ_i. Clearly, $HI_i(y,0) = BI_i(y)$.

In [1], $BI_i(y)$ is determined by creating an artificial task τ_i' with a *best-case computation time* $BC_i' = y$, and applying the same analysis as for best-case response time with constraint deadlines. Furthermore, note that for $HI_i(y,\alpha)$, *delaying* tasks can only preempt τ_i' in the interval $[t_s, t_e - \alpha)$. Therefore, we can extent the analysis for $BI_i(y)$ by considering the minimum number of preemptions that τ_i' can experience due to *delaying* tasks in the interval $[t_s, t_e - \alpha)$. Note that *preemptive* tasks are not influenced by α; hence, the analysis for this part should remain the same as for $BI_i(y)$.

Corollary 1. *The hypothetical best-case interval $HI_i(y,\alpha)$ in which an amount of time $y \in \mathbb{R}^+$ is available for task τ_i is given by the largest $x \in \mathbb{R}^+$ satisfying*

$$x = y + \sum_{h:\pi_h > \theta_i} \left(\left\lceil \frac{x}{T_h} \right\rceil - 1 \right)^+ BC_h + \sum_{d:\theta_i \geq \pi_d > \pi_i} \left(\left\lceil \frac{x-\alpha}{T_d} \right\rceil - 1 \right)^+ BC_d, \tag{1}$$

where the notation w^+ stands for $\max(w,0)$.

To calculate $HI_i(y,\alpha)$, we can use an iterative procedure starting with an upper-bound. The following upper-bound can be used as initial value for such an iterative procedure.

Lemma 1. *An appropriate initial value for the iterative procedure to determine $HI_i(y,\alpha)$ is*

$$HI_i^{(0)}(y,\alpha) = \frac{y}{1-BU_{i-1}}. \tag{2}$$

Proof. Similar to the proof of Lemma 3 in [1]. □

3.2 Hypothetical shortest response time

In this section, we first introduce the notion of *hypothetical shortest response time* and subsequently formulate a number of lemmas related with such a notion.

Definition 2. *The hypothetical shortest response time $\Psi_i(\alpha)$ with $\alpha \in \mathbb{R}^+ \cup \{0\}$ is defined as the shortest response time of a job k of τ_i assuming that delaying tasks activated after or at time $f_{i,k}-\alpha$ cannot start. Furthermore, delaying tasks activated before $f_{i,k}-\alpha$ can preempt τ_i.*

Note that Definition 2 assumes that *delaying* tasks can also preempt a task τ_i. Hence, this definition is similar to the notion of *best-case response time* for FPPS. The only difference is that, for the *hypothetical shortest response time* $\Psi_i(\alpha)$ of a job k of τ_i, delaying tasks activated after $f_{i,k}-\alpha$ cannot start. Therefore, in order to determine $\Psi_i(\alpha)$, we can simply extent the analysis for *best-case response time* for FPPS given in [1] to consider the restriction on *delaying* tasks as follows.

Theorem 1. *Let all tasks of a task-set $\mathcal{T}$ be strictly periodic and let wl_i exist. The hypothetical shortest response time $\Psi_i(\alpha)$ of a task $\tau_i \in \mathcal{T}$ is given by*

$$\Psi_i(\alpha) = \max_{1\le k\le wl_i} (HI_i(k\cdot BC_i,\alpha)-(k-1)T_i). \tag{3}$$

We now present two lemmas related with the notion of *hypothetical shortest response time*.

Lemma 2. *A lower bound for the hypothetical shortest response time $\Psi_i(\alpha)$ is the shortest hold time H_i^{lb} of τ_i when considering only preemptive tasks, i.e $\Psi_i(\alpha) \ge H_i^{lb}$ for all $\alpha \in \mathbb{R}^+ \cup \{0\}$.*

Proof. The proof follows directly from the definition of *hypothetical shortest response time*. Since $\Psi_i(\alpha)$ is always affected by its *preemptive* tasks, the shortest hold time when considering only *preemptive* tasks can never be greater than the shortest response time of a task τ_i. □

Lemma 3. *The best case response time BR_i of a task τ_i is bounded from below by the hypothetical shortest response time $\Psi_i(BR_i)$, i.e. $BR_i \ge \Psi_i(BR_i)$.*

Proof. Let k^{bcrt} be a job of τ_i with $R_{i,k^{bcrt}} = BR_i$. Note that the response time of job k^{bcrt} is not affected by delaying tasks activated after $f_{i,k^{bcrt}} - HT_{i,k^{bcrt}}$ because such tasks cannot preempt τ_i. Now assume that delaying tasks activated after or at time $f_{i,k^{bcrt}} - BR_i$ cannot start. Since $BR_i \ge HT_{i,k^{bcrt}}$, clearly the *best-case response time* cannot increase.

From Definition 2, we now observe that $\Psi_i(BR_i)$ is the shortest response time of a job k of τ_i when assuming that delaying tasks activated after or at time $f_{i,k} - BR_i$ do not start. Based on this definition, and the previous observations, we conclude that $\Psi_i(BR_i)$ can at most be equal to BR_i. Hence, $BR_i \ge \Psi_i(BR_i)$. □

3.3 Best-case response time lower bound

Theorem 2. *Let all tasks of a task-set $\mathcal{T}$ be strictly periodic and let wl_i exist. A lower bound BR_i^{lb} for the best-case response time of τ_i is given by*

$$BR_i^{lb} = \min_{H_i^{lb} \leq \alpha \leq \Psi_i(H_i^{lb})} (\max\{\alpha, \Psi_i(\alpha)\}), \tag{4}$$

where H_i^{lb} is the shortest hold time of τ_i when considering only preemptive tasks.

Proof. We first show that $\min_{H_i^{lb} \leq \alpha \leq \infty} (\max\{\alpha, \Psi_i(\alpha)\})$ is a lower bound for the *best-case response time* of τ_i. Since we are looking for the minimum result of $\max\{\alpha, \Psi_i(\alpha)\}$ considering the interval $H_i^{lb} \leq \alpha \leq \infty$, it is sufficient to show that at least for one value of α within that interval the result of $\max\{\alpha, \Psi_i(\alpha)\}$ is a lower bound for the *best-case response time*. Let $\alpha = BR_i$, clearly this value is in the interval $H_i^{lb} \leq \alpha \leq \infty$. Therefore, we only have to prove that $BR_i \geq \max\{BR_i, \Psi_i(BR_i)\}$. Using Lemma 3, it holds that $\max\{BR_i, \Psi_i(BR_i)\} = BR_i$; hence, concluding the first part of the proof.

We now want to show that the minimum of $\max\{\alpha, \Psi_i(\alpha)\}$ can never be found for $\alpha > \Psi_i(H_i^{lb})$. In order to prove this, first note that by choosing $\alpha = H_i^{lb}$ and using Lemma 2, it holds that $\max\{H_i^{lb}, \Psi_i(H_i^{lb})\} = \Psi_i(H_i^{lb})$. Since for any value of $\alpha > \Psi_i(H_i^{lb})$ the result of $\max\{\alpha, \Psi_i(\alpha)\}$ is always greater than $\Psi_i(H_i^{lb})$, we conclude that the minimum can never be found for such values; therefore, a proper upper bound for α is $\Psi_i(H_i^{lb})$. □

3.4 An algorithm for computing the BCRT lower bound

Theorem 2 gives a lower bound for the *best-case response time* of a task τ_i. However, since $\alpha \in \mathbb{R}^+ \cup \{0\}$ is a continuous variable, the set of values where $H_i^{lb} \leq \alpha \leq \Psi_i(H_i^{lb})$ holds may be infinite. In order to solve this, Algorithm 1 shows a procedure to derive the lower bound of the *best-case response time* presented in Theorem 2 for a task τ_i. Nevertheless, instead of trying all possible values of α in the interval $H_i^{lb} \leq \alpha \leq \Psi_i(H_i^{lb})$, Algorithm 1 starts with $\alpha = H_i^{lb}$ and only increases α on discrete steps that may decrease the result of $\max\{\alpha, \Psi_i(\alpha)\}$. If the result of $\max\{\alpha, \Psi_i(\alpha)\}$ starts increasing instead of decreasing, the algorithm terminates and the minimum is found. [1]

The procedure in Algorithm 1 first assigns to α the hold time of τ_i when considering only preemptive tasks. In Line 3, $\max\{\alpha, \Psi_i(\alpha)\}$ with the initial value of α is chosen as an initial candidate for the best-case lower bound value BR_i^{lb}. Furthermore, in case that BR_i^{lb} is higher than α, it means that it is possible to increase α hopefully leading to a reduction in the best-case lower bound. Hence, Lines 5-7 increase the value of α. More precisely, the job k^{tight} of task τ_i that induces some blocking on the start time of the job experiencing the optimal instant is determined in Line 5. Moreover, Lines 6 and 7 determine the time the activation α of delaying tasks should be pushed to an earlier moment in order to prevent job k^{tight} from inducing some blocking. At this point, if the new value of α exceeds the current BR_i^{lb}, then the result of $\max\{\alpha, \Psi_i(\alpha)\}$ can only increase the value of BR_i^{lb}; hence, the algorithm terminates. Otherwise, it assigns the result of $\max\{\alpha, \Psi_i(\alpha)\}$ to BR_i^{lb} in Line 9 and the process is repeated.

Table 5 shows an overview of the variables used in Algorithm 1.

[1] Note: Probably to show that this is indeed a good point to stop the algorithm, I have to explain first (in a possible proof) that $\Psi_i(\alpha)$ is a monotonically decreasing function

Table 5: Terminology.

Name	Descriptions
BR_i^{lb}	Lower bound of the best-case response time of τ_i.
α	Hypothetical activation of delaying tasks relative to the optimal instant. It is assumed that delaying jobs released at or after α do not start.
k^{tight}	Job of τ_i that blocks the start time of the job experiencing the optimal instant.
DI_i	Length of the time interval between the activation α of delaying tasks and the activation of the job k^{tight} of τ_i.

Algorithm 1 Algorithm to derive a lower bound for the *best-case response time* of task τ_i.

1: **procedure** *bcrtLowerBound*($\mathcal{T}$,i)
2: $\alpha \leftarrow$ shortest hold time of τ_i when considering only preemptive tasks;
3: $BR_i^{lb} \leftarrow \max\{\alpha, \Psi_i(\alpha)\}$;
4: **while** $BR_i^{lb} > \alpha$ **do**
5: $k^{tight} \leftarrow$ the smallest k with $1 \le k \le wl_i$ that leads to BR_i^{lb};
6: $DI_i \leftarrow BR_i^{lb} + (k^{tight}-1)T_i - \alpha$;
7: $\alpha \leftarrow \min_{d:\theta_i \ge \pi_d > \pi_i} (DI_i \mod T_d) + \alpha$;
8: **if** $BR_i^{lb} > \alpha$ **then**
9: $BR_i^{lb} \leftarrow \max\{\alpha, \Psi_i(\alpha)\}$;
end if
end while
10: **return** BR_i^{lb};

3.5 An example

Consider the set of tasks $\mathcal{T}_6$ with characteristics as described in Table 6. We will apply Algorithm 1 to derive a lower bound BR_4^{lb} for the *best-case response time* of task τ_4. Figure 9 shows an example of how each variable of the algorithm would be represented in a schedule after the first iteration. Furthermore, note that tasks τ_1 and τ_2 are *preemptive* tasks of τ_4, whereas τ_3 is a *delaying* task. As can be seen, $\alpha^{(0)} = 22$ represents the initial value of α, and it is initially equal to the hold time of the last job of τ_4. Furthermore, the job of τ_4 activated at time $t = 560$ denoted as k^{tight} is the one that leads to the *hypothetical response time* of $\Psi_4(\alpha^{(0)}) = 36$, and subsequently to $BR_4^{lb(0)} = 36$.

Note that, in Figure 9, it is possible to push the activations of *delaying* task τ_3 to an earlier moment in time in order to allow more space for the execution of τ_4, assuming that the last job of τ_3 cannot interfere with the last job of τ_4. Therefore, the algorithm increments the relative activation α of delaying tasks till an activation of τ_3 coincides with the activation of job k^{tight} of τ_4. This time increment can be calculated as $(DI_4 \mod T_3)$, and it leads to $\alpha^{(1)} = 26$. Note that, at this point, $BR_4^{lb(0)}$ is still larger than $\alpha^{(1)}$; hence, the algorithm will recalculate BR_4^{lb}.

Table 6: Task set $\mathcal{T}_6$.

	T_i	$WC_i = BC_i$	π_i	θ_i	wl_i	WR_i	BR_i
τ_1	35	5	4	4	1	5	5
τ_2	35	5	3	3	1	10	5
τ_3	50	20	2	2	2	62	20
τ_4	70	22	1	2	5	66	27

The *least common multiple* of the periods is 350 and $U^{\mathcal{T}_6} = 1$.

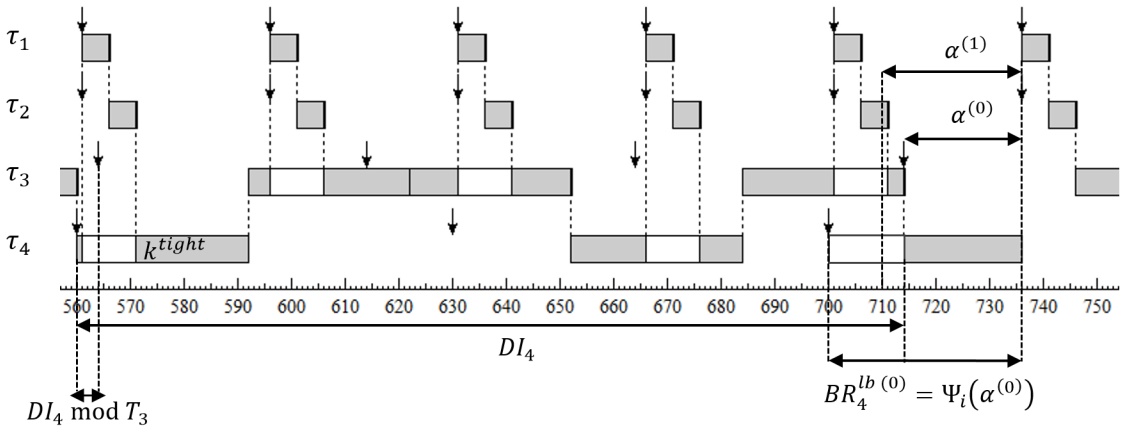

Figure 9: Example after the first iteration of $bcrtLowerBound(\mathcal{T}_6, 4)$.

Figure 10 shows the schedule in the second iterations of $bcrtLowerBound(\mathcal{T}_6, 4)$. As can be seen, the activations of the *delaying* task τ_3 was pushed to an earlier moment in time, allowing to decrease the *hypothetical response time* $\Psi_4(\alpha) = 22$ of the last job. At this point $BR_4^{lb(1)} = \alpha^{(1)} = 26$; hence, the algorithm terminates because there is no previous job restricting the response time of the last job of τ_4.

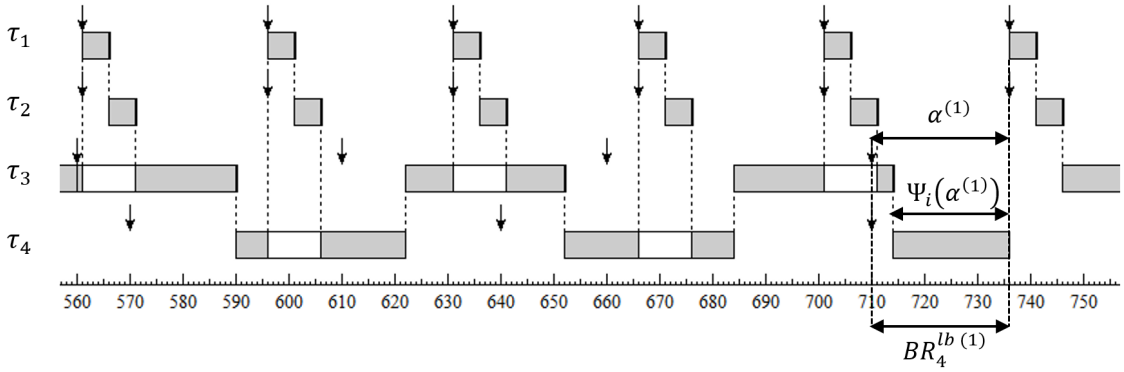

Figure 10: Example after the second iteration of $bcrtLowerBound(\mathcal{T}_6, 4)$.

We conclude that $BR_4^{lb} = 26$ is a proper lower bound for the *best-case response time* of τ_4.

4 Exact best-case analysis for FPTS

4.1 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 τ_h with $\pi_h > \theta_i$, and the set of *delaying* tasks τ_d with $\theta_i \geq \pi_d > \pi_i$. 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 τ_h into the set of *preemptive* tasks that give rise to extra preemptions denoted as τ_e, and the remaining preemptive tasks that we call *best preemptive* tasks denoted as τ_p.

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.

Definition 3. *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 τ_p. Furthermore, the activation of all delaying tasks τ_d and all extra preemptive tasks τ_e, 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 of τ_i is empty when the preemption-threshold of τ_i is equal to its priority. Furthermore, the set of *extra preemptive* tasks τ_e of τ_i is always empty when there are no *delaying* tasks. Therefore, we conclude that the optimal instant for FPTS defined above specializes to an optimal instant for FPPS when $\theta_i = \pi_i$ for a task τ_i.

Figure 11 shows an example of an optimal instant under FPTS for task τ_i of the task-set $\mathcal{T}_7$ described in Table 7. As can be seen, the activation of the *best preemptive* task τ_p coincides with the completion of task τ_i. Furthermore, task τ_e and τ_d are activated an instant δ after the start of τ_i.

Table 7: Task set $\mathcal{T}_7$.

	T_i	$WC_i = BC_i$	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 $U^{\mathcal{T}_7} = 1$.

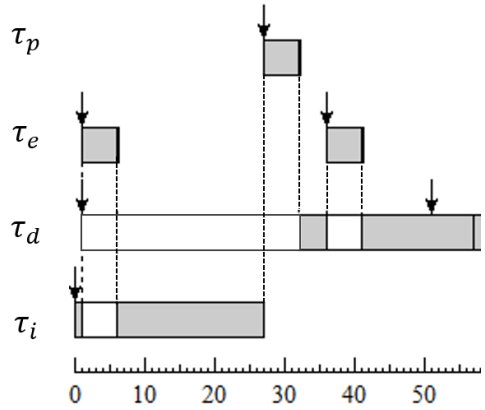

Figure 11: Optimal instant for task τ_i under FPTS.

4.2 Worst, best and general case interval for the execution of a task τ_i

Definition 4. *The worst-case interval $WI_i(i,\mathcal{E})$ is defined as the length of the largest interval in which an amount of time $y \in \mathbb{R}^+$ is available for the execution of τ_i in a task-set $\mathcal{E}$ given that all jobs in the interval assume their best-case computation times.*

$WI_i(y,\mathcal{E})$ is given by the smallest $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{e:\pi_e > \theta_i, \tau_e \in \mathcal{E}} \left\lceil \frac{x}{T_e} \right\rceil BC_e. \tag{5}$$

Definition 5. *The best-case interval $BI_i(i,\mathcal{T})$ is defined as the length of the shortest interval in which an amount of time $y \in \mathbb{R}^+$ is available for the execution of τ_i in a task-set $\mathcal{T}$.*

Function $BI_i(y,\mathcal{T})$ returns the largest positive solution of $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{h:\pi_h > \theta_i, \tau_h \in \mathcal{T}} \left(\left\lceil \frac{x}{T_h} \right\rceil - 1 \right)^+ BC_h. \tag{6}$$

We now define function $HT_i(\mathcal{E})$ 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 $\mathcal{E}$. 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.

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

$$HT_i(\mathcal{E}) = \beta_p + \beta_e + BC_i, \tag{7}$$

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

$$\begin{aligned} \beta_e &= WI_i(\beta_p + BC_i, \mathcal{E}) - \beta_p - BC_i, \\ \beta_p &= BI_i(\beta_e + BC_i, \mathcal{P}) - \beta_e - BC_i. \end{aligned} \tag{8}$$

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.

Definition 6. *Given a job $\iota_{i,k}$ of a task τ_i with a hold time $H_{i,k} = HT_i(\mathcal{E})$, where $\mathcal{E}$ is a set of extra preemptive tasks, the general best-case interval $GI_i(y,\mathcal{E})$ 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.*

$GI_i(y,\mathcal{E})$ is given by the largest $x \in \mathbb{R}^+$ satisfying

$$x = y + \sum_{p:\pi_p > \theta_i, \tau_p \notin \mathcal{E}} \left(\left\lceil \frac{x}{T_p} \right\rceil - 1 \right)^+ BC_p + \sum_{e:\tau_e \in \mathcal{E}} \left(\left\lfloor \frac{x-\alpha}{T_e} \right\rfloor + \left\lceil \frac{\alpha}{T_e} \right\rceil \right) BC_e + \sum_{d:\theta_i \geq \pi_d > \pi_i} \left\lfloor \frac{x-\alpha}{T_d} \right\rfloor BC_d, \tag{9}$$

where $\alpha = HT_i(\mathcal{E})$.

4.3 Best-case response times

Theorem 3. *Let all tasks of a set $\mathcal{T}$ be strictly periodic and let wl_i exist. Furthermore, let $\mathcal{H}_c = \{\tau_h | \pi_h > \theta_i, \exists \tau_d : \theta_i \geq \pi_d > \pi_i\}$ 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*

$$BR_i = \min_{\mathcal{E} \in \hat{E}} \left(\max_{1 \leq k \leq wl_i} (GI_i(k \cdot BC_i, \mathcal{E}) - (k-1)T_i) \right), \tag{10}$$

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

4.4 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 $\mathcal{H}_c$ and a first candidate for the *best-case response time* BR_i. If $\mathcal{H}_c$ 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 $\mathcal{H}_c$ 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*[2]. 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*[3]; hence, the set of possible extra preemptive tasks $\mathcal{H}_c$ 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 2 Exhaustive algorithm to derive *best-case response time* under FPTS.

1: **procedure** *bcrtFPTS*($\mathcal{T}$,i)
2: $(BR_i, \mathcal{H}_c) \leftarrow bcrtInit(\mathcal{T}, i)$;
3: **if** $\mathcal{H}_c \neq \{\}$ **then**
4: $\hat{\mathcal{E}} \leftarrow$ set of all possible combinations of $\mathcal{H}_c$;
5: **for each** $\mathcal{E}$ **of** $\hat{\mathcal{E}}$ **do**
6: $BR_i \leftarrow \min\{BR_i, \max_{1 \leq k \leq wl_i} (GI_i(k \cdot BC_i, HT_i(\mathcal{E}), \mathcal{E}) - (k-1)T_i)\}$;
end for
end if
7: **return** BR_i;

Algorithm 3 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** $\alpha = GI_i(BC_i, \alpha, \{\})$ **then**
4: $BR_i^{init} \leftarrow \max_{1 \leq k \leq wl_i} (GI_i(k \cdot BC_i, \alpha, \{\}) - (k-1)T_i)$;
5: $k^{tight} \leftarrow$ the smallest k with $1 \leq k \leq wl_i$ that leads to BR_i^{init};
6: $DI_i \leftarrow BR_i^{init} + (k^{tight} - 1)T_i - \alpha$;
7: $\alpha \leftarrow \min\{\infty, \min_{d:\theta_i \geq \pi_d > \pi_i} (DI_i \mod T_d)\} + \alpha$;
8: **if** $\alpha \geq BR_i^{init}$ **then**
9: $\mathcal{H}_c = \{\}$;
10: **else**
11: $\mathcal{H}_c = \{\tau_h | \pi_h > \theta_i, BC_h < BR_i - \alpha, \tau_h \in \mathcal{T}\}$
end if
12: **else**
13: $BR_i^{init} \leftarrow \infty$;
14: $\mathcal{H}_c = \{\tau_h | \pi_h > \theta_i, \tau_h \in \mathcal{T}\}$;
end if
15: **return** $BR_i^{init}, \mathcal{H}_c$;

[2]Note: Right now this is just a claim, based on my observations and intuition. Not sure how to support this formally, yet.
[3]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.