

# 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

May 23, 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  $\tau_i$  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  $\tau_i$  can be affected by a higher priority task that cannot preempt  $\tau_i$ .

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.

	$T_i$	$WC_i = BC_i$	$\pi_i$	$\theta_i$	$wl_i$
$\tau_1$	35	10	3	3	1
$\tau_2$	50	20	2	2	2
$\tau_3$	70	22	1	2	5

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

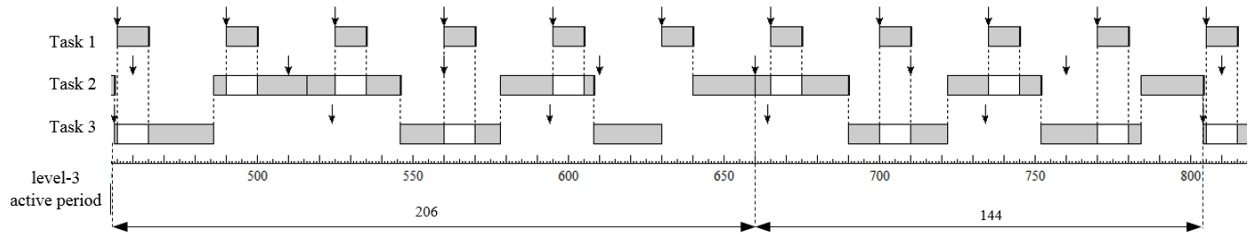


Figure 1: The shortest response time of task  $\tau_3$  occurs in the first job of the first level-3 active period.

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

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  $\tau_i$ .

## 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  $\tau_i$  coincides with the completion of a job of  $\tau_i$ .*

Consider the taskset presented in Table 2. In Figure 2 a schedule is shown where the activation of higher priority tasks  $\tau_1$  and  $\tau_2$  coincides with the completion of a job of  $\tau_4$  at time  $t = 736$ . As can be seen, the shortest response time for  $\tau_4$  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  $\tau_4$ . Figure 3 shows a schedule for  $\mathcal{T}_2$  where a response time of  $R_4 = 27$  for task  $\tau_4$  is found at time  $t = 709$ .

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

	$T_i$	$WC_i = BC_i$	$\pi_i$	$\theta_i$	$wl_i$	$BR_i$
$\tau_1$	35	5	4	4	1	5
$\tau_2$	35	5	3	3	1	5
$\tau_3$	50	20	2	2	2	20
$\tau_4$	70	22	1	2	5	27

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

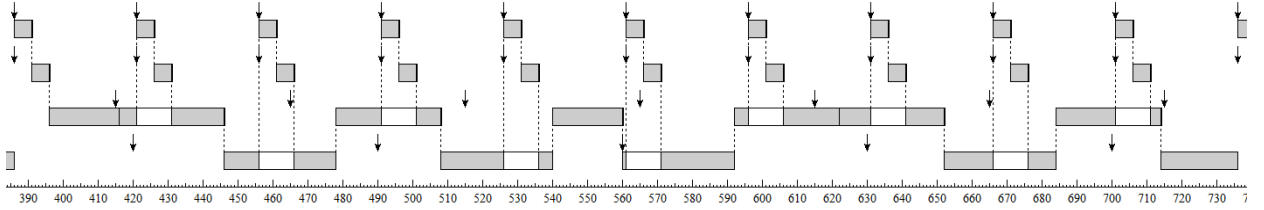


Figure 2: Simultaneous release of preemptive tasks coincides with the completion of a job of task  $\tau_4$  at time 736. The phases used to construct this schedule are  $\phi_1 = \phi_2 = 1$ ,  $\phi_3 = 15$ , and  $\phi_4 = 0$ .

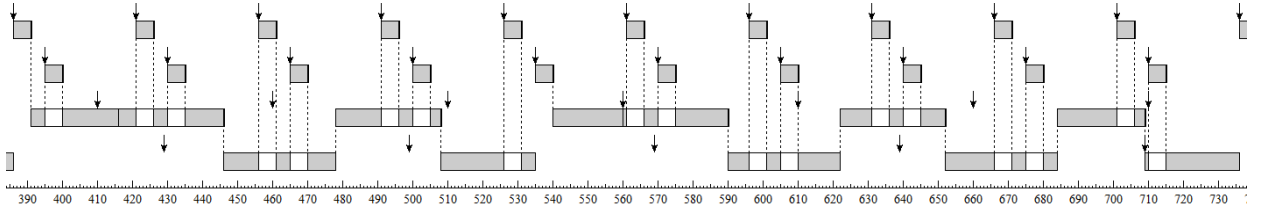


Figure 3: Schedule for  $\mathcal{T}_2$  where the *best-case response time* ( $BR_4 = 27$ ) for  $\tau_4$  is assumed at time  $t = 709$ . The phases used to construct this schedule are  $\phi_1 = 1$ ,  $\phi_2 = \phi_3 = 10$ , and  $\phi_4 = 9$ .

**Fact 6.** *A higher priority task that cannot preempt a task  $\tau_i$  can provoke that the completion of task  $\tau_i$  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  $\tau_1$  and  $\tau_3$  of  $\mathcal{T}_3$  where the activation of the highest priority task  $\tau_1$  coincides with the completion of task  $\tau_3$ . However, this situation cannot be assumed when considering also  $\tau_2$ . Figure 5 shows that as soon as  $\tau_2$  is introduced in the schedule, task  $\tau_3$  will always experience one preemption by  $\tau_1$ . Note that the first job of  $\tau_2$  in Figure 5 delays the start time of the first job of  $\tau_3$ ; hence, causing the completion of  $\tau_3$  to not occur simultaneously with the activation of  $\tau_1$ .

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

	$T_i$	$WC_i = BC_i$	$\pi_i$	$\theta_i$	$wl_i$	$WR_i^T$	$BR_i^T$	$RJ_i^T$	$WR_i^P$	$BR_i^P$	$RJ_i^P$
$\tau_1$	80	20	3	3	1	20	20	0	20	20	0
$\tau_2$	30	15	2	2	8	104	15	89	35	15	20
$\tau_3$	240	50	1	2	1	120	70	50	230	165	65

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

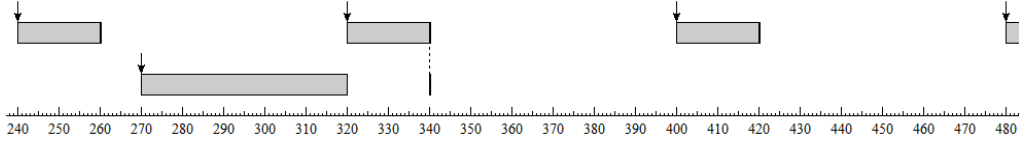


Figure 4: Schedule of task-set  $\mathcal{T}_3 \setminus \{\tau_2\}$ . The phases used to construct this schedule are  $\phi_1 = 0$  and  $\phi_3 = 30$ .

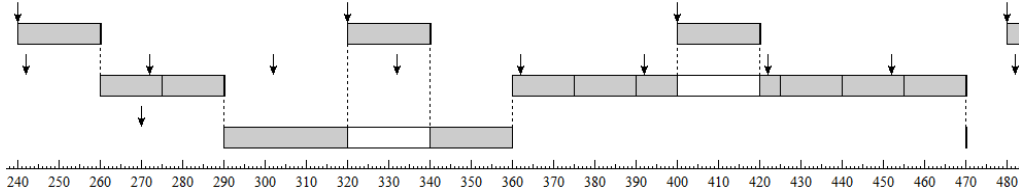


Figure 5: Schedule of task-set  $\mathcal{T}_3$ . The phases used to construct this schedule are  $\phi_1 = 0$ ,  $\phi_2 = 2$  and  $\phi_3 = 30$ .

**Fact 7.** Under FPTS and deadlines at most equal to periods, the best-case response time of a task  $\tau_i$  is not necessarily equal to the best-case response time when considering only higher priority tasks that can preempt  $\tau_i$ .

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

	$T_i$	$WC_i = BC_i$	$\pi_i$	$\theta_i$	$wl_i$	$BR_i$	$WR_i$
$\tau_1$	18	9	3	3	1	9	17
$\tau_2$	24	8	2	3		8	24
$\tau_3$	45	7	1	2		12	38

The least common multiple of the periods is 360 and  $U^{\mathcal{T}_4} \approx 0.99$ .

## 2.1 Blocking tasks

**Fact 8.** A lower priority task that can block the execution of a task  $\tau_i$  cannot affect the best-case response time of  $\tau_i$ .

Consider task-set  $\mathcal{T}_5$  described in Table 5. Figure 6 shows an schedule for such a task-set, but only considering tasks  $\tau_1$  and  $\tau_2$ . As can be seen, the best-case response time  $BR_2 = 50$  for task  $\tau_2$  is assumed in the schedule. Introducing the blocking task  $\tau_3$  could affect the response time of task  $\tau_2$  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  $\mathcal{T}_5$  scheduled in such a way that the best-case response time for task  $\tau_2$  is assumed.

Table 5: Task set  $\mathcal{T}_5$ .

	$T_i$	$WC_i = BC_i$	$\pi_i$	$\theta_i$	$wl_i$	$BR_i$
$\tau_1$	80	20	3	3	1	20
$\tau_2$	240	50	2	2	1	50
$\tau_3$	30	15	1	2	8	15

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

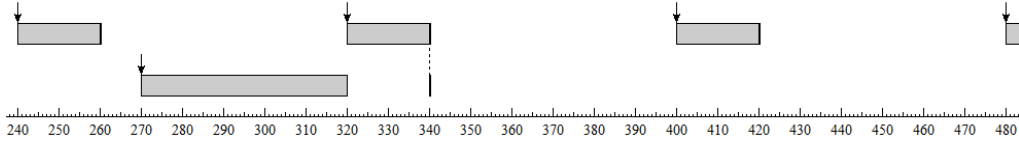


Figure 6: Schedule of taskset  $\mathcal{T}_5 \setminus \tau_3$ . The phases used to construct this schedule are  $\phi_1 = 0$  and  $\phi_2 = 30$ .

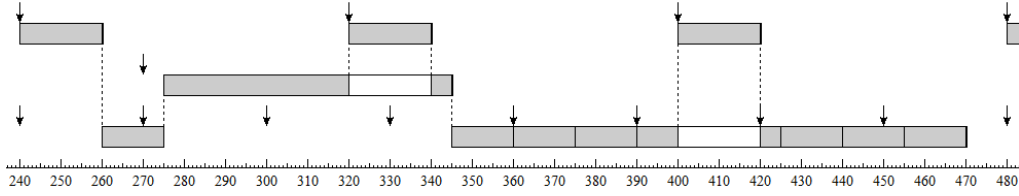


Figure 7: Schedule of task-set  $\mathcal{T}_5$ . The phases used to construct this schedule are  $\phi_1 = \phi_3 = 0$  and  $\phi_2 = 30$ .

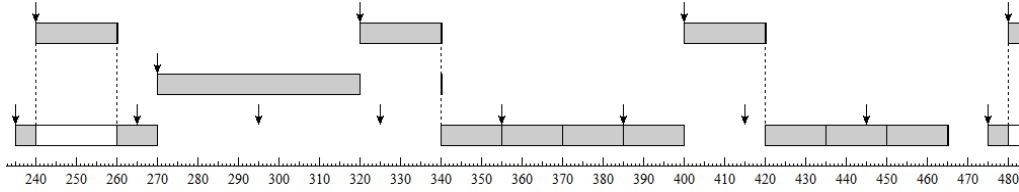


Figure 8: Schedule of task-set  $\mathcal{T}_5$  where the *best-case response time* for task  $\tau_2$  is assumed. The phases used to construct this schedule are  $\phi_1 = 0$ ,  $\phi_3 = 25$  and  $\phi_2 = 30$ .

### 3 BCRT lower bound for FPTS

#### 3.1 Hypothetical best-case interval for execution of a task $\tau_i$

**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  $\tau_i$  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  $\tau_i$ .

Note that  $HI_i(y, \alpha)$  assumes that all higher priority tasks can preempt  $\tau_i$ . Therefore,  $HI_i(y, \alpha)$  can be seen as an extension of the *best-case interval*  $BI_i(y)$  defined in [1] for FPPS, 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  $\tau_i$ . Clearly,  $HI_i(y, 0) = BI_i(y)$ .

In [1],  $BI_i(y)$  is determined by creating an artificial task  $\tau'_i$  with a *best-case computation time*  $BC'_i = y$ , and applying the same analysis as for best-case response time with constraint deadlines

for FPPS. Furthermore, note that for  $HI_i(y, \alpha)$ , *delaying* tasks can only preempt  $\tau'_i$  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  $\tau'_i$  can experience due to *delaying* tasks in the interval  $[t_s, t_e - \alpha)$ . Note that *preemptive* tasks are not influenced by  $\alpha$ ; 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  $\tau_i$  is given by the largest  $x \in \mathbb{R}^+$  satisfying*

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

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*

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

*Proof.* Similar to the proof of Lemma 3 in [1].  $\square$

## 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  $\tau_i$  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  $\tau_i$ .*

Note that Definition 2 assumes that *delaying* tasks can also preempt a task  $\tau_i$ . 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  $\tau_i$ , delaying tasks activated after or at  $f_{i,k} - \alpha$  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*

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

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  $\tau_i$  when considering only preemptive tasks, i.e.  $\Psi_i(\alpha) \geq H_i^{lb}$  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  $\tau_i$ .  $\square$

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

*Proof.* Let  $k^{bert}$  be a job of  $\tau_i$  with  $R_{i,k^{bert}} = BR_i$ . Note that the response time of job  $k^{bert}$  is not affected by delaying tasks activated after  $f_{i,k^{bert}} - HT_{i,k^{bert}}$  because such tasks cannot preempt  $\tau_i$ . Now assume that delaying tasks activated after or at time  $f_{i,k^{bert}} - BR_i$  cannot start. Since  $BR_i \geq HT_{i,k^{bert}}$ , 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  $\tau_i$  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,  $BR_i \geq \Psi_i(BR_i)$ .  $\square$

### 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  $\tau_i$  is given by*

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

where  $H_i^{lb}$  is the shortest hold time of  $\tau_i$  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  $\tau_i$ . 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  $\alpha$  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  $\alpha$  is  $\Psi_i(H_i^{lb})$ .  $\square$

### 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  $\tau_i$ . 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  $\tau_i$ . Nevertheless, instead of trying all possible values of  $\alpha$  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  $\alpha$  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.<sup>1</sup>

The procedure in Algorithm 1 first assigns to  $\alpha$  the hold time of  $\tau_i$  when considering only preemptive tasks. In Line 3,  $\max\{\alpha, \Psi_i(\alpha)\}$  with the initial value of  $\alpha$  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  $\alpha$ , it means that it is possible to increase  $\alpha$  hopefully leading to a reduction in the best-case lower bound. Hence, Lines 5-7 increase the value of  $\alpha$ . More precisely, the job  $k^{tight}$  of task  $\tau_i$  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  $\alpha$  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  $\alpha$  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 6 shows an overview of the variables used in Algorithm 1.

<sup>1</sup>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 6: Terminology.

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

---

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

---

```

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

```

---

### 3.5 An example

Consider the set of tasks  $\mathcal{T}_7$  with characteristics as described in Table 7. We will apply Algorithm 1 to derive a lower bound  $BR_4^{lb}$  for the *best-case response time* of task  $\tau_4$ . 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  $\tau_1$  and  $\tau_2$  are *preemptive* tasks of  $\tau_4$ , whereas  $\tau_3$  is a *delaying* task. As can be seen,  $\alpha^{(0)} = 22$  represents the initial value of  $\alpha$ , and it is initially equal to the hold time of the last job of  $\tau_4$ . Furthermore, the job of  $\tau_4$  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  $\tau_3$  to an earlier moment in time in order to allow more space for the execution of  $\tau_4$ , assuming that the last job of  $\tau_3$  cannot interfere with the last job of  $\tau_4$ . Therefore, the algorithm increments the relative activation  $\alpha$  of delaying tasks till an activation of  $\tau_3$  coincides with the activation of job  $k^{tight}$  of  $\tau_4$ . This time increment can be calculated as  $(DI_4 \bmod T_3)$ , 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 7: Task set  $\mathcal{T}_7$ .

	$T_i$	$WC_i = BC_i$	$\pi_i$	$\theta_i$	$wl_i$	$WR_i$	$BR_i$
$\tau_1$	35	5	4	4	1	5	5
$\tau_2$	35	5	3	3	1	10	5
$\tau_3$	50	20	2	2	2	62	20
$\tau_4$	70	22	1	2	5	66	27

The *least common multiple* of the periods is 350 and  $U^{\mathcal{T}_7} = 1$ .

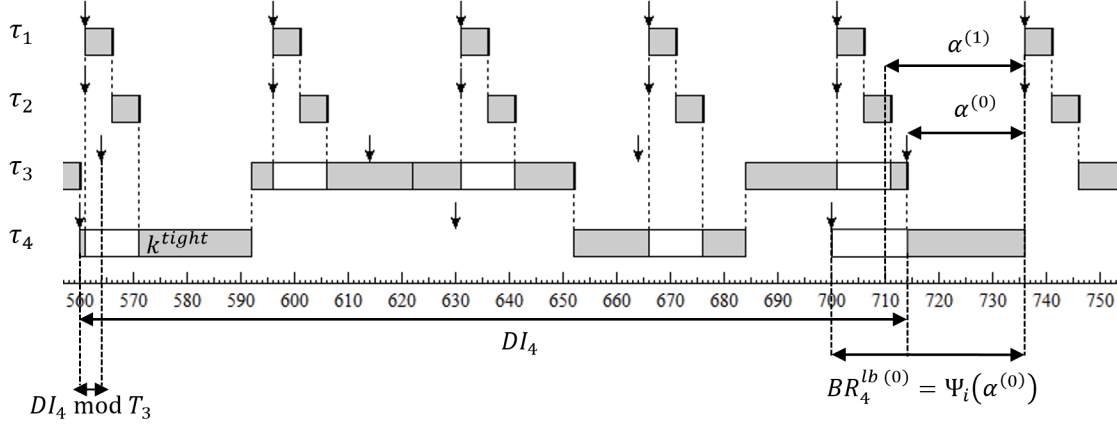


Figure 9: Example after the first iteration of  $bcrtLowerBound(\mathcal{T}_7, 4)$ .

Figure 10 shows the schedule in the second iterations of  $bcrtLowerBound(\mathcal{T}_7, 4)$ . As can be seen, the activations of the *delaying* task  $\tau_3$  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  $\tau_4$ .

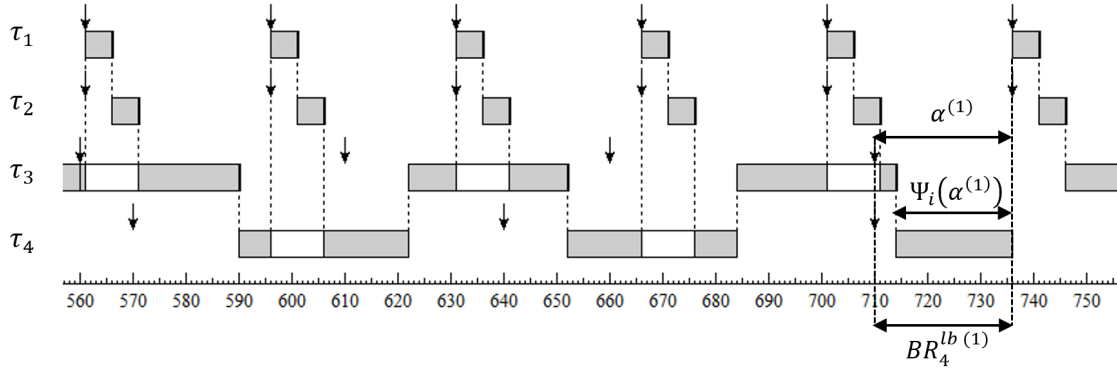


Figure 10: Example after the second iteration of  $bcrtLowerBound(\mathcal{T}_7, 4)$ .

We conclude that  $BR_4^{lb} = 26$  is a proper lower bound for the *best-case response time* of  $\tau_4$ .

## 4 Exact best-case analysis for FPTS

### 4.1 Optimal instant

Based on Fact 3, we can 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 of *preemptive* tasks  $\tau_h$  with  $\pi_h > \theta_i$ , and the set of *delaying* tasks  $\tau_d$  with  $\theta_i \geq \pi_d > \pi_i$ . Furthermore, Fact 5 shows that for some cases the *best-case response time* of a task  $\tau_i$  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  $\tau_i$  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  $\tau_h$  into the set of *preemptive* tasks that give rise to extra preemptions denoted as  $\tau_e$ , and the remaining preemptive tasks that we call *best preemptive* tasks denoted as  $\tau_p$ .



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  $\tau_i$ .

**Definition 3.** In FPTS, an optimal instant for a task  $\tau_i$  occurs when the completion of a job  $\iota_{i,k}$  of  $\tau_i$  coincides with a simultaneous activation of all best preemptive tasks  $\tau_p$ . Furthermore, the activation of all delaying tasks  $\tau_d$  and all extra preemptive tasks  $\tau_e$ , coincides with  $s_{i,k} + \delta$ , where  $\delta$  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  $\tau_i$  is empty when the preemption-threshold of  $\tau_i$  is equal to its priority. Furthermore, the set of *extra preemptive* tasks  $\tau_e$  of  $\tau_i$  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  $\tau_i$ .

Figure 11 shows an example of an optimal instant under FPTS for task  $\tau_i$  of the task-set  $\mathcal{T}_8$  described in Table 8. As can be seen, the activation of the *best preemptive* task  $\tau_p$  coincides with the completion of task  $\tau_i$ . Furthermore, task  $\tau_e$  and  $\tau_d$  are activated an instant  $\delta$  after the start of  $\tau_i$ .

Table 8: Task set  $\mathcal{T}_8$ .

	$T_i$	$WC_i = BC_i$	$\pi_i$	$\theta_i$
$\tau_p$	35	5	4	4
$\tau_e$	35	5	3	3
$\tau_d$	50	20	2	2
$\tau_i$	70	22	1	2

The least common multiple of the periods is 350 and  $U^{\mathcal{T}_8} = 1$ .

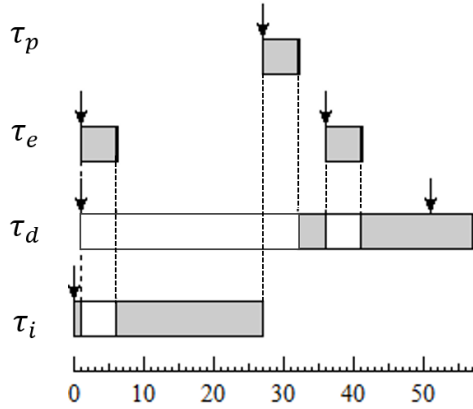


Figure 11: Optimal instant for task  $\tau_i$  under FPTS.

## 4.2 Worst, best and general case interval for the execution of a task $\tau_i$

**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  $\tau_i$  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

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

**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  $\tau_i$  in a task-set  $\mathcal{T}$ .

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

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

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  $\delta$  is a sufficiently small amount of time.

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

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

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  $\beta_p$  and  $\beta_e$  are given by the smallest solution of the following recursive equations:

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

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

**Definition 6.** Given a job  $\iota_{i,k}$  of a task  $\tau_i$  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  $\tau_i$ .

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

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

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  $\tau_i$  is given by

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

where  $\hat{\mathcal{E}}$  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  $\tau_i$  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  $bcrInit(\mathcal{T}, i)$  procedure. Line 2 first assigns to  $\alpha$  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*<sup>2</sup>. 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  $\alpha$  of *delaying* tasks has to be increased in order to reduce the response time. If  $\alpha$  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*<sup>3</sup>; 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  $bcrFPTS(\mathcal{T}, i)$ 
2:    $(BR_i, \mathcal{H}_c) \leftarrow bcrInit(\mathcal{T}, i);$ 
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:        $BR_i \leftarrow \min\{BR_i, \max_{1 \leq k \leq wl_i} (GI_i(k \cdot BC_i, HT_i(\mathcal{E}), \mathcal{E}) - (k-1)T_i)\};$ 
7:     end for
8:   end if
9:   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  $bcrInit(\mathcal{T}, i)$ 
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:      $\alpha \leftarrow \min\{\infty, \min_{d: \theta_i \geq \pi_d > \pi_i} (DI_i \bmod T_d)\} + \alpha;$ 
8:     if  $\alpha \geq BR_i^{init}$  then
9:        $\mathcal{H}_c = \{\};$ 
10:    else
11:       $\mathcal{H}_c = \{\tau_h | \pi_h > \theta_i, BC_h < BR_i - \alpha, \tau_h \in \mathcal{T}\}$ 
12:    end if
13:  else
14:     $BR_i^{init} \leftarrow \infty;$ 
15:     $\mathcal{H}_c = \{\tau_h | \pi_h > \theta_i, \tau_h \in \mathcal{T}\};$ 
16:  end if
17:  return  $BR_i^{init}, \mathcal{H}_c$ ;
```

---

<sup>2</sup>Note: Right now this is just a claim, based on my observations and intuition. Not sure how to support this formally, yet.

<sup>3</sup>Note: Probably this has to be proved as well.

## 4.5 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  $\tau_i$  is assumed when some *preemptive* tasks are activated a sufficiently small amount of time after the start of the job  $k^{best}$  of  $\tau_i$  that assumes the best-case response time. We denote this category of tasks as *extra preemptive* because they give rise to extra preemptions in job  $k^{best}$ , whereas the *preemptive* tasks that interfere at least as possible with  $k^{best}$  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  $\tau_i$ . The set of *best preemptive* tasks is simply given by  $\mathcal{P}_i = \mathcal{H}_i - \mathcal{E}_i$ . 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  $\tau_i$  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 9 shows an example of such a nontrivial task-set. As we will show, determining the best-case response time of  $\tau_i$  for this and similar cases is NP-hard.

Table 9: Task set  $\mathcal{T}_9$ .

task	$T_i$	$WC_i = BC_i$	$\pi_i$	$\theta_i$
$\{\tau_{h1}, \tau_{h2}, \tau_{h3}, \tau_{h4}, \tau_{h5}\}$	$\{35, \dots, 35\}$	$\{3.3, 2.3, 2, 1.3, 1.1\}$	$\{7, \dots, 3\}$	$\{7, \dots, 3\}$
$\tau_d$	50	20	2	2
$\tau_i$	70	22	1	2

The least common multiple of the periods is 350 and  $U^{\mathcal{T}_9} = 1$ .

Figure 12 shows a timeline for  $\mathcal{T}_9$  where all *preemptive* tasks  $\mathcal{H}_i$  are activated simultaneously with the completion of the last job  $\iota_{i,1}$  of  $\tau_i$  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  $\tau_i$  because the job  $\iota_{i,3}$  activated at time  $t = -176$  starts upon activation. Hence, pushing the activation of  $\tau_i$  would modify the schedule. In order to reduce the response time of  $\iota_{i,1}$ , the activations of  $\tau_d$  should be pushed to an earlier moment in time till one of its activations occurs at time  $t = -176$ . In this way,  $\tau_d$  will prevent job  $\iota_{i,3}$  to start upon activation and, consequently, it will be possible to reduce  $R_{i,1}$ . However, forcing  $\tau_d$  to be activated at time  $t = -176$  would lead to an activation of  $\tau_d$  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  $H_{i,1} > 26$ .

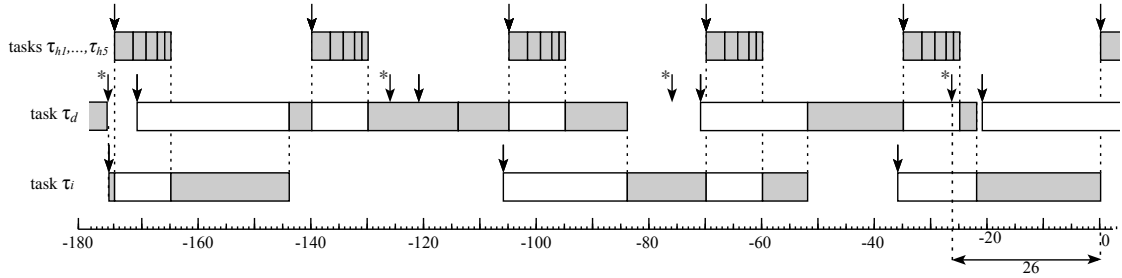


Figure 12: A timeline for  $\mathcal{T}_9$  when the set of *extra preemptive* tasks  $\mathcal{E}_i$  is empty. Down-arrows with a "star" denote the activations for  $\tau_d$  that would allow to reduce the response time of  $\tau_i$ .

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  $\mathcal{T}_9$ , the solution of this problem is depicted in Figure 13, where the *extra preemptive* tasks that minimize the response time of  $\tau_{i,1}$  are  $\tau_{h_2}$  and  $\tau_{h_3}$ . Therefore, the best-case response time of task  $\tau_i$  is  $BR_i = BC_i + BC_{h_2} + BC_{h_3} = 26.3$ .

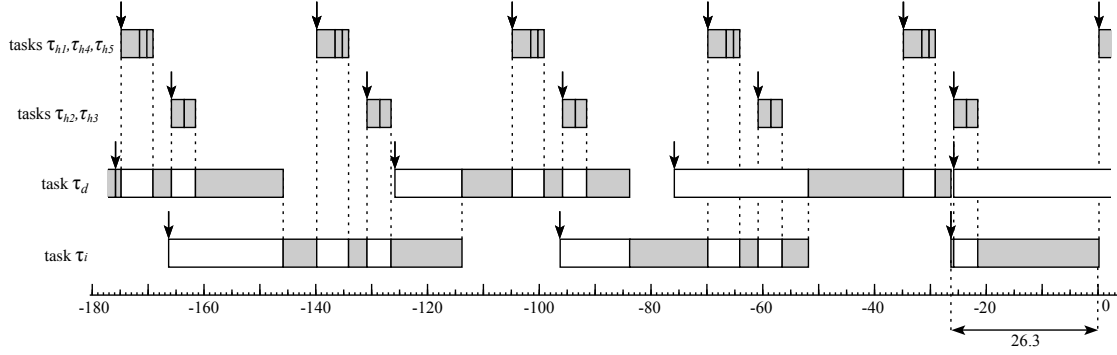


Figure 13: A timeline for  $\mathcal{T}_9$  showing the best-case response time of  $\tau_i$  with  $BR_i = 26.3$ . The set of *extra preemptive* tasks that leads to the best-case response time is  $\mathcal{E}_i = \{\tau_{h_2}, \tau_{h_3}\}$ .

## 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.